A Framework for Organization-Aware Agents

Jensen, Andreas Schmidt; Dignum, Virginia; Villadsen, Jørgen

Published in:
Autonomous Agents and Multi-Agent Systems

Link to article, DOI:
10.1007/s10458-015-9324-2

Publication date:
2017

Document Version
Peer reviewed version

Autonomous Agents and Multi-Agent Systems First online: 12 January 2016
The final publication is available at Springer via http://dx.doi.org/10.1007/s10458-015-9324-2

A Framework for Organization-Aware Agents

Andreas Schmidt Jensen · Virginia Dignum · Jørgen Villadsen

Abstract Open systems are characterized by the presence of a diversity of heterogeneous and autonomous agents that act according to private goals. Organizations, such as those used in real-life to structure human activities such as task allocation, coordination and supervision, can regulate the agents' behavior space and describe the expected behavior of the agents. Assuming an open environment, where agents are developed independently of the organizational structures, agents need to be able to reason about the structure, so that they can deliberate about their actions and act within the expected boundaries and work towards the objectives of the organization.

In this paper, we present the AORTA reasoning framework and show how it can be integrated into typical BDI-agents. We provide operational semantics that enables agents to make organizational decisions in order to coordinate and cooperate without explicit coordination mechanisms within the agents. The organizational model is independent of that of the agents, and the approach is not tied to a specific organizational model, but uses an organizational metamodel. We show how AORTA helps agents work together in a system with an organization for choosing the best tender for a building project.

Keywords Organization-aware agents · Organizational reasoning · Operational semantics

1 Introduction

In multi-agent systems, intelligent agents interact with each other and the environment in order to achieve their goals. They are flexible in that they can react to changes and proactively pursue objectives, and they are autonomous, enabling them to decide by themselves what to do. They are usually characterized by cognitive elements, such as beliefs, desires, goals and planning. In open systems, where agents can enter and exit freely, it is hard to predict and control the agents' behavior. Such systems include, e.g., auctions or webshops,

A. S. Jensen · J. Villadsen
DTU Compute, Technical University of Denmark, Kgs. Lyngby, Denmark
E-mail: ascje@dtu.dk; jovi@dtu.dk

V. Dignum
Faculty of Technology, Policy and Management, Delft University of Technology, Delft, The Netherlands
E-mail: m.v.dignum@tudelft.nl

where the agents entering the system need to know the protocols used in order to participate. By allowing agents to understand these protocols, it is possible to make generic agents that can enter different systems, instead of specific agents for each case. Such systems are often specified by means of organizational models (similarly to what is done in human societies), which defines (amongst other things) roles that describe the expected behavior of the agents in the system, and the interactions between those roles.

An organizational model provides by itself only *structure*: roles are specified and are related to different expectations according to the requirements or goals of the organization. As such, it provides no central control over the agents and no way to ensure the fulfillment of the expectations. To further complicate things, agents joining the organization may be designed by distinct designers, which altogether leads to uncertainty about the agents' motives and actions.

An organization usually has a set of control mechanisms available, which makes it possible to better manage the agents entering the organization. The type of control exercised in an organization is often divided into two categories: *regimentation* and *regulation* [31]. Regimented systems are characterized by the fact that certain actions are restricted, whereas regulated systems puts no such constraints on actions, but rather makes an attempt to ensure agents will not violate the constrains, despite being able to do so [23]. This is often done by imposing sanctions on violating agents [4, 40]. In reality, the control mechanisms are often designed such that certain parts are regimented, while others are regulated. Consider, e.g., a conference management system: the program chair may restrict the submission of papers after the submission deadline (regimentation), while deciding to sanction papers not following the formatting instructions (regulation).

Whereas strong guarantees can be made about regimented systems (e.g., a forbidden action is never successfully executed), the same is more difficult for regulated systems. It might be possible to guarantee that upon violation of certain expectations, a sanction is always put in effect, but given the uncertainty of the agents in the organization, it is not possible to guarantee that 1) the sanction recovers the system and 2) the behavior of agent is affected by the sanction.

Since open systems do not in general pose restrictions upon the agents entering, it seems that in order to ensure that nothing undesirable happens, two options are possible: (1) make no assumptions about the agents' organizational reasoning capabilities and regiment all interaction between agent and system (similar to governors in AMELI/ISLANDER [21]), or (2) assume that the agents are able to reason about the expectations of the organization and are able to handle sanctions. The first option limits the actions the agent can take and ensures that nothing bad happens, but since the agents are not free to do as they like (since some actions are restricted), the execution of the system may not be very efficient. The second option lets the agents do what they want (within reason), but they may be sanctioned if they violate the expectations. However, if agents that do not understand the organizational model enter the system, they will first of all not be able to understand what is expected of them, but secondly, if (and when) they violate the expectations, they may not even understand the sanctions, which may then have no effect.

Our motivation for this work is as follows: in open systems, anyone can participate, but the success of an agent's participation depends on its ability to understand and reason about the system – including an organization, if present. We therefore look at it the other way round: we want anyone to be able to *successfully* participate in open systems. Successful participation is a vague statement, but our goal is clear: provide previously "organization-ignorant" agents with capabilities that allow them to perform organizational reasoning, such that they understand the organizational model of the system, can reason about participation

and are able to choose whether or not to comply with the expectations that stem from the role(s) they enact. Such capabilities will essentially make agents *organization-aware* [40].

In this paper, we present a complete version of the operational semantics of the AORTA[1] reasoning framework [29, 30], which provides cognitive agents with a component that enable them to reason about organizational models and to act upon them. In [29], we first presented AORTA, and in [30], we provided an implementation of the framework, integrated with *Jason*. This paper extends the AORTA framework as follows:

- *Obligations*: We incorporate conditional obligations with deadlines, allowing agents to deliberately violate objectives and be subject to sanctioning.
- *Capabilities*: The framework supports the notion of agent capabilities, making it possible to suggest which roles match the agent best.
- *Semantics*: We present the complete operational semantics of AORTA.

AORTA is agent-centered, and assumes that the organization is preexisting and independent from the agent, thus focusing on letting the agent reason about the organization. By separating the organization from the agent, the architecture of the agent is independent from the organizational model, and the agent is free to decide on how to use AORTA in its reasoning. On the other hand, having a separate component for organizational reasoning may complicate reasoning about multiple organizations in a single system. Furthermore, as is usually the case with generic systems, there will be systems in which a specialized framework is better suited. We claim, however, that for most purposes, having a generic framework, such as a AORTA, makes the agent able to participate within organizations without impeding its behavior too much.

The separation of AORTA and the agent is achieved by basing the reasoning on rules using an organizational metamodel, designed to support different organizational models. The metamodel is based on roles, objectives and obligations, which are common traits of many organizational models. As a proof of concept, we provide a translation from OperA organizational models to the AORTA metamodel to demonstrate its usefulness, and show how this enables previously "organization-ignorant" agents to use an OperA organizational model to successfully participate in a system.

The rest of the paper is organized as follows. In Section 2, we discuss what is meant by organizational reasoning. In Section 3, we describe how organizational reasoning is done in AORTA by describing the different phases in the AORTA reasoning cycle, and we present the AORTA organizational metamodel, a model based on roles, objectives and obligations, which is used to represent organizational models inside the framework. In Section 4, we present the operational semantics of AORTA. We provide an initial evaluation of the framework in Section 5. Finally, we conclude the paper by discussing future work in Section 6.

2 Organizational Reasoning

Agents that are able to perform organizational reasoning are called *organization-aware* agents [40]. The concept of organizational reasoning covers many aspects: reasoning about entering and exiting an organization, reasoning about which roles to enact, whether to comply or violate certain norms and how to coordinate tasks with other members of the organization. In the following, we define what we mean by organizational reasoning and compare our approach to related work on organizational reasoning. Considering the three dimensions of organizational reasoning presented in [40], our work can be positioned as follows:

[1] AORTA stands for Adding Organizational Reasoning to Agents.

- Direction of organizational reasoning: AORTA takes a top-down approach, assuming an existing model and providing agents with the means to reason about this model.
- Understanding organizational specifications: AORTA provides agents with the possibility to understand organizational specifications, and assumes that agents understand the domain ontology used in the organizational model.
- Phases of participation: AORTA provides reasoning rules that allow agents to successfully move through the phases of participation: entering the organization, playing roles in the organization and leaving the organization.

2.1 Understanding organizational specifications

In order to understand an organizational specification, two things are required. First, the agent should understand the concepts used to describe the organization (i.e., how is a role specified, what are the objectives, etc.), and second, the agent should be able to operationalize the specification based on this knowledge. That is, given that the agent understands the notion of a role, how does it decide to enact that role, and how does the agent commit to completing organizational objectives alongside its own, personal objectives. This process assumes that agents understand the domain ontology and the modelling ontology (i.e the agent is able to understand what a role is, and shares the domain ontology used by the organizational model. As discusssed in [14], this is not a trivial aspect, but is one that can be determined by design of agents entering the organization.

Much effort has been put into adding such capabilities to agents [6, 8, 9, 15, 16, 28, 33], and the focus has usually been on agents that are based on the belief-desire-intention (BDI) model. The approach has most commonly been to modify the BDI interpreter loop [38] to incorporate organizational matters into the reasoning process. While not dealing specifically with organizational reasoning, recent work by Wallace and Rovatsos [43] proposes a mechanism for social reasoning that is separate from the agent's BDI reasoning.

In [6] conflicts between beliefs, obligations, intentions and desires are discussed with a focus on a distinction between *internal* conflicts, e.g. contradictory beliefs, conflicting obligations and *external* conflicts, such as a desire which is in conflict with an obligation. The solution proposed, the BOID architecture, imposes a strict ordering between beliefs, obligations, intentions and desires, such that the order of derivation determines the agent's attitude. Thus different agent types emerge; an agent deriving desires before beliefs is a wishful-thinking agent, while an agent deriving obligations before desires is social. Another approach is discussed in [15] in which the orderings are mapped into a common scale, such that very desirable situations could outweigh the cost of violating certain obligations. Such ordering should be quite dynamic since, for example, obligations toward a trusted agent should become less important if that agent becomes less trustworthy.

In [16], obligations are represented using Prohairetic Deontic Logic [42], a preference-based dyadic deontic logic which allows for contrary-to-duty obligations (obligations holding in a sub-ideal context). Furthermore, they propose a modified BDI-interpreter in which selected events are augmented with potential deontic events, which, put simply, are obligations and norms that may become applicable when choosing a plan.

While these, and other approaches, do enable agents to reason about organizational matters, they do so by changing the architecture of the agents. By doing so, existing implementations of agents may not work as expected using such architecture, meaning that in order to impose organizational reasoning capabilities upon such agents, changes in seemingly unrelated matters may be necessary. Furthermore, we believe that by abstracting the

cognitive reasoning away from the organizational reasoning, we benefit from the principle of *separation of concerns*, which is often mentioned as a key point for using organizations in multi-agent systems: *what* should be done is separated from *how* it should be done [22].

This is accommodated by the MOISE$^+$ model, which separates organized multi-agent systems into a system level and an agent level. The MOISE$^+$ model is based on three organizational *dimensions*: the structural, functional and deontic dimensions [28]. The system level, S-MOISE$^+$, provides an interface (a middleware) between the agents and the organization using a special agent, the OrgManager, to change the organizational state, ensuring organizational consistency. The agent level, J-MOISE$^+$, joins *Jason* and MOISE$^+$, by making organizational actions available to agents, such that they can reason about (and change, using the OrgManager) an organization. J-MOISE$^+$differs from AORTA in that the organization-oriented reasoning is done as a part of the agent's normal reasoning process, whereas agents using AORTA perform the organizational reasoning inside the AORTA component, and then decide how to complete their objectives at a different level. The main advantage of keeping the reasoning apart in AORTA is that it allows agents on different agent platforms to perform the same kind of organizational reasoning without any extra development required.

A generic architecture for organization-aware agents is suggested in [40], where the agent is divided into two layers: a cognitive layer, which provides more or less the same reasoning capabilities as seen in, e.g. BDI agents, and an organizational layer, which communicates with the organization and handles decision making that has to do with the organization, e.g. reasoning about role enactment. Our approach is similar to this; agents are enriched with an organizational reasoning component, which performs organizational reasoning *separate from* the cognitive reasoning performed by the agent. Our framework is generic enough to be useful for many different kinds of cognitive agents, such that they gain organizational reasoning capabilities without making changes to the "normal" reasoning.

2.2 Phases of participation

We divide participation into three phases: *entering*, *playing roles* and *leaving*. In the following, we discuss work done by others in relation to these phases, and later, when we have presented the operational semantics of AORTA, we show how some of these concepts can be used in AORTA to participate in an organization.

Entering an organization An agent entering an organization has to consider several things before choosing to play a role. In general, the agent has to decide whether it wants to enter the organization. This includes, first of all, considering *why* it should enter it. What can the agent gain from entering the organization, which capabilities are gained, and what resources will be accessible. Furthermore, what capabilities are lost, and what does the organization expect from the agent, once it enters. A considerable amount of work has been done in the area of deciding which role(s) to adopt in an organization. In [11, 18], agent goals and role objectives are compared in order to decide whether the agent is compatible and consistent with a role. A role is compatible with an agent if the objectives of the role are a subset of the goals of the agent, and the objectives can be achieved using the agent's plans. An agent and a role are consistent if the objectives and goals are not in conflict. Agents can use this to decide how well it matches a role, and what is gained (e.g., the role provides plans for achieving the agent's goals) or lost (a goal is in conflict with an objective) by enacting a given role. Furthermore, the papers distinguish between different kinds of role enactment,

such as social or selfish enactment, which describes how the agent gives priority to the objectives of the role and its own goals.

It is also possible to consider the relationship between agents enacting roles in an organization before deciding whether to participate [7]. By distinguishing between what an agent can do, has access to, and is allowed to do, they define what it means to have power over resources. Then, by defining social dependencies, it is possible to define a power relation between agents in an organization. Before entering the organization, the agents can use this to reason about what (or who) they will have power over, and who (if any) will have power over them.

Finally, the agents should consider their own capabilities before choosing a role to enact. That is, what can the agent do, and what capabilities are required for successfully enacting a given role. In order to consider the agent's capabilities, we should first make clear what kinds of capabilities an agent can have. A straightforward definition of agent capability is the set of states the agent can achieve. This has been investigated in the context of BDI logic in [34], where BDI systems with a plan library are considered. Here, having a capability for ϕ means having at least one plan with the goal ϕ as its trigger. In [1, 39], the notion of capability is extended to include actions the agent is able to perform, percepts the agent can perceive and expressions the agent can utter. In [1], the OperA model is extended to include role capabilities, and agents entering the organization have to match the capabilities in order to enter. This decision is done by a gatekeeper, which decides, based on what the agent tells the gatekeeper that it can do, whether or not the agent is allowed to participate. In [39], it is investigated how agents can reason about their own capabilities using reflection.

In AORTA a goal is a desired state, i.e. a state the agent desires to achieve. Therefore, we will define a agent capability as a partial state description ϕ that an agent is able to achieve.

Playing roles in an organization Once an agent has entered the organization, it has to reason about the organization's expectations based on the role(s) it enacts. That is, the organization expects the agent enacting a role to achieve the objectives of the role, so the agent has to reason about 1) how to complete the objectives, 2) how this relates to the completion of its own goals and 3) what happens if it, for some reason, it has to or, decides to violate some of the expectations. There is, of course, some overlap of reasoning between this phase and the previous, since this kind of reasoning played a part in deciding to enact the role.

In order to reason about completing organizational objectives, the agent has to understand what is required to complete an objective, and whether it can do so itself or by delegating the objective to other agents. The first point is concerned with relating the objectives of the organization to its own goals. The second point requires the agent to understand the organizational model, such that it can deduce hierarchical relations between different roles, and use this to decide how objectives can be delegated.

The organizational expectations might not be met by the agent, and in such case, the agent should know that. Depending on the agents ability, different levels of understanding the model emerge: it might be able to understand before executing an action that this will violate an obligation, or it might just be able to understand that a obligation has now been violated. Furthermore, the agent should ideally be able to understand the sanctions that might be imposed upon the agent. If not, it has no way to decide whether or not to violate an obligation, since it does not know the impact a sanction may have. The B-DOING framework [15] incorporates the possibility to reason about sanctions by introducing the *worth* of a state and *cost* of violating an obligation. This is used to make a preference ordering of obligations, such that agents can choose to violate obligations when the worth of the resulting state outranks the sanction. In [33], norms are perceived by the agent in the environment,

and can then choose to accept or reject it. By rejecting, sanctions are imposed, thus reasoning about norms and sanctions in this case is done at different levels. First, when perceiving a norm, the agent decides to accept or reject a norm. Then, if it accepts the norm, it changes its behavior to conform to the norm. However, the change in behavior may not show until later, once the agent chooses to complete the objective described by the norm.

Leaving an organization The agent may have different reasons for wanting to leave an organization. Just as it had to decide why to enter the organization, it has to decide why it wants to leave. These decisions are, of course, related. If, for example, the agent entered the organization in order to achieve a specific objective, it seems perfectly reasonable to want to leave the organization once that objective has been completed. Once the agent has decided to leave the organization (see [26] for reasons for leaving an organization), it has to deact the roles it enacts in the organization. By deacting a role, the agent tells the organization to drop the expectations about the agent, but there might be situations where this is not ideal. For example, in an auction house, if the agent tries to deact the buyer role before having paid for the items it has bought, it should generally not be allowed to do so; at least not without getting sanctioned as well [40].

It might also be the case that the agent is thrown out of the organization (e.g. as a sanction), because it repeatedly violates obligations. This is only possible when enactment and deactment is determined by the whole community [12], since other agents in the organization have to decide that the agent in question should be removed from the organization.

2.3 Norms

We now turn to *norms*, which describe what role-enacting agents are expected to do (or not do) from the perspective of the organization. Norms are used in organizational modeling to specify the expectations about the enactors, while still allowing them to be deliberately violated. We noted above that agents should be able to understand the organization's expectations and also reason about the result of violating those expectations. In the following, we discuss how norms can be modelled, and how this makes it possible for the agents to reason about obligations and violations.

Deontic logic is often used as a formalization of obligations, permissions and prohibitions. In deontic logic, it is possible to state, e.g. $O\phi$, meaning that ϕ *ought to be the case*. Deontic logic has been applied as a formalism for multi-agent systems [41], and provides mechanisms to reason about norms and violation thereof.

We take the common view [2, 15, 33] that an obligation specifies an ideal *state* of a system, from the point of view of the creator of the obligation (the organization, in our case). An agent cannot be obliged to *do* ϕ; it will rather be obliged to *achieve* ψ (e.g., by doing ϕ, by taking some other action(s) that lead to ϕ, or by making sure ψ is achieved by some other means, such as making some another agent do it). By looking at states rather than actions, we abstract away from specific actions. This corresponds well to the fact that the same state often can be achieved in different ways, so rather than requiring agents with a specific ability to *do* something, the fulfillment of an obligation requires that the agent *achieves* something. This choice means that we cannot distinguish between different ways of achieving the same state, but since we take the point of view of the organization, this is not a problem, since the focus here is on the results and not on the specific steps taking to achieve the results.

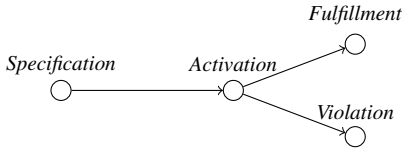


Fig. 1 The states of an obligation. Once specified, the obligation is activated when the activation condition holds. It is fulfilled when the desirable state is achieved, and it is violated if the deadline is reached before it has been fulfilled.

A key aspect of obligations is that of violation: the organization does not constrain the agents to fulfill their obligations. Instead, it is necessary to detect the violation of an obligation, making it possible to punish agents that achieves an undesirable state. However, detecting when an obligation has been violated can be difficult. Detecting violation of a prohibition is straightforward: once we reach the prohibited state, the prohibition has been violated. If we apply the same method to obligations, we run into a problem: either the obligation is violated until it is fulfilled (since the ideal state is not reached yet), or the obligation is violated only when the ideal state cannot be reached anymore. The first option is not very intuitive. Imagine borrowing a book from a friend; this creates the obligation to return the book at some point, however we would probably not immediately say that we have violated the obligation. On the other hand, if we set the book on fire, the obligation is clearly violated, since it can no longer be returned, but in other cases it might not be as straightforward. f

If we cannot detect if an obligation has been violated, it becomes difficult to punish agents that does not follow the rules. If the agents have no incentives for following the rules, it becomes very difficult to ensure fulfillment of the organization's objectives. In this paper, we therefore consider conditional obligations with a deadline:

$$O(p < \delta \mid c),$$

where p should be achieved, the condition c specifies the state in which an obligation becomes active and the deadline δ specifies the first state in which the obligation has been violated. That is, once the deadline holds and p has not been achieved, the obligation is violated. The states of an obligation are shown in Figure 1. We assume that an obligation can only be activated once and that once an obligation has been fulfilled, it can never be violated. The last point implies that the fulfillment of an obligation happens in the state where p holds, and at that point, the deadline becomes irrelevant, since it can never be violated.

The activation of an obligation refers to the activation of a *ground* obligation. When we define the organizational metamodel in Section 3.4, the conditional obligation can be seen as a template with variables. This means that there can be multiple activated instantiations of one obligation template, but we assume that each ground obligation can only be activated once. Our work can be extended to release the assumptions, but that is out of scope for this paper.

Even though we say that fulfillment of an obligation is based on what the agent *achieves*, this does not mean that our semantics is based on *stit*-logic [3]. If an obligation to achieve p is only fulfilled once the agent sees to it that p holds, it may actually violate the obligation if another agent independently achieves p. For example, it would be counterintuitive to be punished for not returning the book, if someone else has returned it.

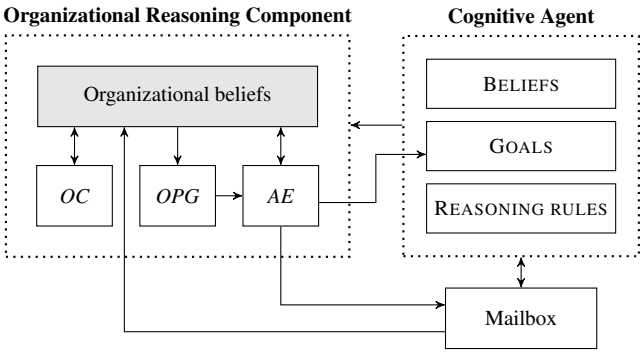


Fig. 2 The AORTA component. The arrows indicate flow of information. Obligations and options are generated from the organizational beliefs, and actions are based on the generated options.

3 The AORTA Reasoning Framework

The AORTA reasoning framework adds a component to cognitive agents, which provides them with organizational reasoning capabilities. It assumes a preexisting organization, is independent from the agent, and focuses on reasoning rules that specify how the agent reasons about the organization. The organization is completely separated from the agent, meaning that the architecture of the agent is independent from the organizational model, and the agent is free to decide on how to use AORTA in its reasoning. The separation is possible because AORTA is based on an organizational metamodel, to which other organizational models can be translated.

Organizational reasoning in AORTA divided into three phases: *obligation check* (OC), *option generation* (OPG) and *action execution* (AE). The OPG-phase uses the organizational state to generate possible organizational options. The OC-phase uses the agent's mental state and organizational state to determine if obligations are activated, satisfied or violated, and updates the organizational state accordingly. The agent considers these options in the AE-phase using reasoning rules, which can alter the organizational state, the agent's intentions or send messages to other agents. The component is shown in Figure 2.

The component is connected to certain inner structures of the cognitive agent's reasoning component, e.g., the belief and goal base, and the agent's mailbox. Given that our component is designed to be independent from the cognitive agent, we need to ensure 1) that changing the agent's beliefs (or goals) does not create inconsistencies in the agent's reasoning and 2) that the agent's beliefs (or goals) used in the organizational reasoning does not change during the organizational reasoning phases. In [30], we showed that both of these issues could be addressed by executing the AORTA reasoning cycle and the cognitive agent's reasoning cycle sequentially. Since the two cycles are not executed in parallel, changes to the agent's beliefs, goals or organizational beliefs will only be used (and altered) by a single process. We note that in very active environments it may prove to be inefficient since the agent's reasoning cycle is not executed as often. However, our approach is meant to be generic and it might not be suitable for certain scenarios.

Note that within the AORTA component, the organizational state is actually the agent's *beliefs* about the organizational state. This has a number of implications: first, the agent may hold wrong beliefs about the organization. Second, in order to let others know about organizational actions, certain measures must be taken (e.g., communication can be used

to inform others). Third, there is no central organizational entity, which effectively means that it is the agents in the system that makes up the organization. While the first implication can be considered a disadvantage, we believe that it more accurately corresponds to our idea of agents in an organization. Furthermore, by not having a central organizational entity, different kinds of organizations having different kinds of rules of enactment and deactment can be implemented in the same framework.

We assume the component is connected to a *cognitive* agent, i.e., an agent with mental attitudes (such as beliefs and goals) and practical reasoning rules. This could, for example, be BDI agents (defined by their mental attitudes: beliefs, desires and intentions), which have been used as a basis for many agent programming languages [5, 10, 25].

3.1 Obligation check

The OC-phase is the first phase of the organizational reasoning in AORTA. The agent's state is used to determine for each obligation if it should change to a new state (cf. the states in Figure 1). That is, if the condition for activating an obligation has happened, the framework activates the obligation by updating the organizational state. Similarly, it checks whether the obligation has been satisfied (the objective is completed) or violated (the deadline was reached before the objective was completed).

AORTA has no central mechanism that can detect violations and sanction violating agents. It is up to the agents in the system to do so. One way to handle this is to activate new obligations on the condition of a violation (a so-called contrary-to-duty obligation [42]). For example, an agent that violates the obligation to return a book to the library is then obliged to pay a fine. If the new obligation is violated, a new obligation may be activated.

3.2 Option generation

The OPG-phase uses the mental state of the agent and the organizational state to consider what the agent can do regarding the organization. The following organizational aspects are considered in the OPG-phase:

Role enactment: Roles that are possible to enact given the agent's capabilities.
Role deactment: (Currently enacting) roles that have been fulfilled or are no longer useful.
Obligations: States the agent is currently obliged to achieve.
Delegation: Objectives that can be delegated based on a dependency relation.
Information: Obtained information that other agents will benefit from knowing.

The options that are generated in this phase are then available to act upon in the AE-phase.

3.3 Action execution

The AE-phase uses reasoning rules to decide how to react on a given option in a given context. The AE-phase selects at most one option to act upon. The reasoning is based on rules of the form[2]

$$option : context \rightarrow action$$

[2] Inspired by the plan syntax of AgentSpeak(L) [37].

where *option* is a previously generated option, *context* is a state description that should hold for an action to be applicable, and *action* is the action to be executed.

The agent has actions available to enact or deact a role, commit to or drop an objective, and send messages. This corresponds well to the options that can be generated in the previous phase.

We assume that the agent's actions directly related to the domain of the organization, that is, we do not consider counts-as relations [24] to relate agent actions with organizational objectives.

3.4 AORTA Organizational Metamodel

Organizational models are used in multi-agent systems to give agents an explicit representation of an organization. Similarly to [19] we use concepts from Organizational Theory (OT), which, even though it lacks formality, has been studied for years and has been applied successfully. OT defines an organization as an entity in which individuals have roles, and use these roles to accomplish collective goals. Organizations are furthermore defined with a purpose; individuals in the organization have intentions and objectives, which lead to the overall collective goals.

Different models are proposed in the literature (e.g. MOISE$^+$ [28], OperA [17], ISLANDER [20]). These models typically use concepts from OT as well, especially the notion of *roles*, abstracting implementation details away from expectations, and *objectives*, defining the desired outcome of the organization. Furthermore, organizational models often take a normative view of a system, specifying how the agents ideally should behave, what they are allowed to do, and what is explicitly forbidden.

We base reasoning in AORTA on an organizational metamodel, which is based on roles, objectives and obligations.

Definition 1 (Organizational metamodel) The organizational metamodel of AORTA is defined by the following predicates:

role($Role, Objs$)	*Role* is the role name, and *Objs* is a set of objectives, where each *Obj* is a partial state description.
obj(*Obj*, *SubObjs*)	*Obj* is the name of an objective, and *SubObjs* is a set of sub-objectives.
dep($Role_1, Role_2, Obj$)	Role $Role_1$ depends on role $Role_2$ for completion of objective *Obj*.
rea($Ag, Role$)	Agent *Ag* enacts role *Role*.
cond(*Role*, *Obj*, *Deadline*, *Cond*)	A conditional obligation for role *Role* to complete *Obj* before *Deadline* when *Cond* holds.
obl(*Ag*, *Role*, *Obj*, *Deadline*)	An obligation for agent *Ag* playing role *Role* to complete *Obj* before *Deadline*.
viol(*Ag*, *Role*, *Obj*)	Agent *Ag* playing role *Role* has violated the obligation to complete *Obj*.

A role is defined only by its name and its *main* objectives. Sub-objectives of an objective are specified using obj-predicates. We distinguish between the different states of norms by using different predicates. For example, the conditional obligation

$$O_{reader}(returned(Book) < Deadline \mid borrowed(Book))$$

is represented by the predicate cond(*reader*, *returned*(*Book*), *Deadline*, *borrowed*(*Book*)). If an agent *Bob* enacts the *reader* role and borrows the book "1984", the obligation is activated and we have

$$O_{\mathsf{rea}(bob,reader)}(returned(\text{“}1984\text{”}) < Deadline).$$

This is represented by the predicate obl(*bob*, *reader*, *returned*(*"1984"*), *Deadline*). A violation of the obligation is represented by the predicate viol(*bob*, *reader*, *returned*(*"1984"*)).

While the metamodel can be used as-is, we further argue its usefulness by the fact that existing organizational models can be translated into it. As a proof of concept, we include in Appendix a translation from the OperA organizational model to the metamodel.

4 Operational Semantics of AORTA

In Section 2, we described what we mean by organizational reasoning, and in Section 3, we showed how our framework integrates this into agents using a reasoning component that generates options and executes actions. In this section, we proceed to more formally show how this can be done. First, we define the semantics of organization-aware agents in AORTA using a temporal logic to make clear the intended behavior. Second, we define the operational semantics of executing agents in the AORTA framework. We define the semantics using a transition system [35], which consists of a set of transition rules that define the transformation from one configuration to another.

One of the key ideas of AORTA is the notion of the organizational knowledge base used for reasoning about options and actions. Furthermore, the framework makes use of an *options base* containing the options generated in the OPG-phase.

Definition 2 (Mental state) The AORTA mental state is based on knowledge bases. A knowledge base is a set of predicates believed by the agent to be true. Each knowledge base is based on a predicate language, L, with typical formula ϕ. The agent's belief base and intention base are denoted Σ_a and Γ_a, respectively. The language of the organization is denoted $L^{org} \subseteq L$, and the option language is denoted $L^{opt} \subseteq L$. The organizational state and options are denoted Σ_o and Γ_o, respectively. The mental state, *MS*, is then a tuple of knowledge bases:

$$\text{MS} = \langle \Sigma_a, \Gamma_a, \Sigma_o, \Gamma_o \rangle,$$

where $\Sigma_a, \Gamma_a \subseteq L$, $\Sigma_o \subseteq L^{org}$ and $\Gamma_o \subseteq L^{opt}$. We write **MS** to denote a set of mental states.

The definition above puts the organizational state inside the mental state of each agent. That is, we consider what the agent *believes* about the organization, meaning that it may not be consistent with the actual state of the organization (and different agents may hold different beliefs about the organization). We believe this corresponds well to the idea that agents hold beliefs about the environment, which, similarly, may be wrong.

Definition 3 (Options) The option language, L^{opt} with typical element γ is defined as follows:

$$\gamma ::= \mathsf{role}(R) \mid \neg\mathsf{role}(R) \mid \mathsf{obj}(O) \mid \neg\mathsf{obj}(O) \mid \mathsf{send}(R, ilf, \phi),$$

where R is a role identifier, O is an objective, *ilf* is *tell* or *achieve*, and $\phi \in L$ is a message.

Notice that $\mathsf{send}(R, ilf, \phi)$ specifies an option to send a message to a role R rather than a specific agent. This gives the agent more freedom as to deciding who to inform. Furthermore, at the point in time where the option is generated, the agent might not (yet) know who should receive the message.

Each of the knowledge bases in the mental state can be queried using *reasoning formulas*.

Definition 4 (Reasoning formulas) AORTA uses reasoning formulas, L_R, with typical element ρ, which are based on organizational formulas, option formulas, belief formulas and goal formulas.

$$\rho ::= \top \mid \mathsf{org}(\phi) \mid \mathsf{opt}(\phi) \mid \mathsf{bel}(\phi) \mid \mathsf{goal}(\phi) \mid \neg\rho \mid \rho_1 \wedge \rho_2,$$

where $\phi \in L$.

Organizational formulas, $\mathsf{org}(\phi)$, query the organizational beliefs, option formulas, $\mathsf{opt}(\phi)$, query the options base, belief formulas, $\mathsf{bel}(\phi)$, query the belief base and goal formulas, $\mathsf{goal}(\phi)$, query the goal base. For example, the formula

$$\mathsf{org}(\mathsf{rea}(bob, bidder)) \wedge \mathsf{goal}(bid(RFT, Bid))$$

succeeds if agent *Bob* plays the bidder role and *bid*(*RFT*,*Bid*) is a goal.

Definition 5 (Semantics of reasoning formulas) The semantics are based on the agent's mental state, $\mathrm{MS} = \langle \Sigma_a, \Gamma_a, \Sigma_o, \Gamma_o \rangle$.

$$\begin{array}{ll}
MS \models \top & \\
MS \models \mathsf{bel}(\phi) & \textit{iff } \phi \in \Sigma_a \\
MS \models \mathsf{goal}(\phi) & \textit{iff } \phi \in \Gamma_a \\
MS \models \mathsf{org}(\phi) & \textit{iff } \phi \in \Sigma_o \\
MS \models \mathsf{opt}(\phi) & \textit{iff } \phi \in \Gamma_o \\
MS \models \neg\rho & \textit{iff } MS \not\models \rho \\
MS \models \rho_1 \wedge \rho_2 & \textit{iff } MS \models \rho_1 \text{ and } MS \models \rho_2
\end{array}$$

The above definition makes it possible to reason about the agent's mental state. We can, for example, check who enacts a specific role, what the agent's options are, and what it believes. However, this does not capture the fact that the agents and the organization reside in an environment, which none of them fully control. To capture this, we use the Logic for Agent Organizations (LAO) [19] extended with the ability to deal with obligations.

LAO uses Kripke semantics to describe the system, such that a world in the semantics corresponds to a state of the environment, and a transition corresponds to changes happening in the environment (e.g. agents executing actions). As we are going to use the semantics for formalizing the execution, we let each world correspond to a set of agent mental states, i.e. the agents' view of the environment and organization. This makes the semantics correspond to what the agents believe rather than what may actually be the case (since beliefs can be wrong).

LAO (and our extension, LAO with Obligations, LAO_2) uses the temporal logic CTL* to describe the environment and organizational concepts. The language of LAO_2, $\mathscr{L}$, consists of two subsets: $\mathscr{L}_s$, the set of state formulas with typical element Φ, and $\mathscr{L}_p$, the set of path formulas with typical element ϕ. We define the set of state formulas as follows:

$$\Phi ::= \rho \mid \neg\Phi \mid \Phi \vee \Phi \mid A\phi \mid E\phi,$$

where ϕ is a path formula and ρ is a reasoning formula. The set of path formulas is defined as follows:

$$\phi ::= \Phi \mid \neg\phi \mid \phi \vee \phi \mid F\phi \mid G\phi \mid X\phi \mid \phi U \phi,$$

where the operators are the usual CTL* operators: A is all, E is exists, F is future, G is always in the future, X is next, and U is until.

The semantic structure over which formulas of $\mathscr{L}$ are interpreted is the tuple

$$M = (W, Rt, \pi),$$

where W is a non-empty set of states, Rt is a partial order of states, and π associates each $w \in W$ with a mental state, $\pi : W \to MS$. The semantics distinguish between state and path formulas. State formulas are interpreted wrt. a state $w \in W$ and a path formula is interpreted wrt. a path through the structure given in Rt. A path in Rt is a sequence $(w_i, w_{i+1}, \dots)$, where $w_i, w_{i+1}, \dots \in W$ and $\forall i : (w_i, w_{i+1}) \in Rt$. A path is denoted $rt = (w_0, w_1, \dots)$ and we can refer to a state i in a path using $rt(i)$. For state formulas, we write $M, w \models \phi$ to denote that state formula ϕ is true in M at state w. Similarly, we write $M, rt \models \phi$ for path formulas. We let $paths(W, Rt)$ denote the set of all paths in M.

Definition 6 (Semantics)

$$\begin{array}{lll}
M,w \models p & \text{iff} & \pi(w) \models p, \quad \text{where } p \in L_R \\
M,w \models \neg\phi & \text{iff} & M,w \not\models \phi \\
M,w \models \phi \vee \psi & \text{iff} & M,w \models \phi \text{ or } M,w \models \psi \\
M,w \models A\phi & \text{iff} & \forall \text{rt} \in paths(W,Rt), \text{ if } rt(0) = w \text{ then } M,rt \models \phi \\
M,w \models E\phi & \text{iff} & \exists \text{rt} \in paths(W,Rt), \text{ s.t. } rt(0) = w \text{ and } M,rt \models \phi \\
M,rt \models p & \text{iff} & M,rt(0) \models p, \quad \text{where } p \in L_R \\
M,rt \models \neg\phi & \text{iff} & M,rt \not\models \phi \\
M,rt \models \phi \vee \psi & \text{iff} & M,rt \models \phi \text{ or } M,rt \models \psi \\
M,rt \models F\phi & \text{iff} & \exists i (M,rt(i)) \models \phi \\
M,rt \models G\phi & \text{iff} & \forall i (M,rt(i)) \models \phi \\
M,rt \models X\phi & \text{iff} & M,rt(1) \models \phi \\
M,rt \models \phi U \psi & \text{iff} & \exists i \text{ s.t. } M,rt(i) \models \psi \text{ and } \forall 0 \leq k < i \; M,rt(k) \models \phi
\end{array}$$

The semantics differs from the semantics in LAO in that we use reasoning formulas rather than propositions. A reasoning formula is true in a world iff it is true in the mental state of the agent in that world.

Obligations are first specified, and are then activated once their condition is met. The semantics for obligations include the role for which the obligation applies (when specified), and the role enactment relation (when activated).

Definition 7 (Conditional obligation with deadline) Given a role r and an agent α enacting r, a conditional obligation with deadline, $O_r(p < \delta \mid c)$, is defined as:

$$\begin{array}{lll}
M,rt \models O_r(p < \delta \mid c) & \text{iff} & \forall i (M,rt(i), i \models O_r(p < \delta \mid c)) \\
M,rt \models O_{\mathsf{rea}(\alpha,r)}(p < \delta) & \text{iff} & \forall i (M,rt(i), i \models O_{\mathsf{rea}(\alpha,r)}(p < \delta)) \\
M,rt,i \models O_r(p < \delta \mid c) & \text{iff} & \begin{cases} \exists \alpha \text{ s.t. } (M,rt(i+1) \models c \wedge \mathsf{org}(\mathsf{rea}(\alpha,r)) \text{ and} \\ \quad M,rt(i+1), i+1 \models O_{\mathsf{rea}(\alpha,r)}(p < \delta)), \text{ or} \\ M,rt(i+1) \models \neg c \text{ and} \\ \quad M,rt(i+1), i+1 \models O_r(p < \delta \mid c) \end{cases} \\
M,rt,i \models O_{\mathsf{rea}(\alpha,r)}(p < \delta) & \text{iff} & \begin{cases} M,rt(i+1) \models \neg p \wedge \neg\delta \text{ and} \\ \quad M,rt(i+1), i+1 \models O_{\mathsf{rea}(\alpha,r)}(p < \delta), \text{ or} \\ M,rt(i+1) \models p \wedge \neg\delta \wedge AG\neg viol(\alpha,r,p), \text{ or} \\ M,rt(i+1) \models \neg p \wedge AG(\delta \wedge viol(\alpha,r,p)) \end{cases}
\end{array}$$

where $p, \delta, c \in L_R$. $O_{\mathsf{rea}(\alpha,r)}(p < \delta)$ is an activated instance of the obligation for α enacting role r.

The semantics are divided into conditional obligations and activated obligations. A conditional obligation for a role r can be activated for an agent α when α enacts r and the condition holds. That is the case in worlds w, where $\pi(w) \models \mathsf{org}(\mathsf{rea}(\alpha, r))$. A conditional obligation is then activated in a state where the condition, c, holds and the activated obligation holds. There are four possible outcomes for an activated obligation:

- $\neg p \wedge \neg \delta$: The obligation is still active, since it has not been fulfilled and the deadline is not reached.
- $p \wedge \neg \delta$: The obligation has been fulfilled, and the deadline is not reached, thus the obligation can never be violated.
- $\neg p \wedge \delta$: The deadline is reached without fulfilling the obligation, thus the obligation will always be violated.
- $p \wedge \delta$: If the obligation has been fulfilled and the deadline is reached, the outcome depends on when p occurred. If it occurred before δ, we the obligation can never be violated, otherwise it will always be violated.

The semantics do not take into account infinite deadlines, such as $\bot$. In such case, the deadline will never occur, so the obligation can never be violated.

Note that the obliged state, p, the deadline, δ and the condition, c are reasoning formulas, meaning that we can specify obligations to achieve a state ϕ (i.e. $\mathsf{bel}(\phi)$), but also e.g. an organizational state (i.e. $\mathsf{org}(\phi)$). This creates the possibility to create an obligation to e.g. enact or deact a role. It furthermore enables reorganization, e.g. in the context of LAO [13], to create new roles, dependency relations and objectives, but that is out of scope this paper.

We define organization-specific actions, that are used to perform changes to the organization, or committing to completing objectives.

Definition 8 (Actions) The set of actions with typical element a is denoted Act.

$$a ::= \mathsf{enact}(role) \mid \mathsf{deact}(role) \mid \mathsf{commit}(\phi) \mid \mathsf{drop}(\phi) \mid \mathsf{send}(rcp, msg)$$

Agents can decide when to execute a given action based on action rules. Action rules match options generated in the OPG-phase, and are applicable if the context follows from the agent's mental state.

Definition 9 (Action rule) The set of action rules is defined by:

$$R_A = \{o \ : \ ctx \rightarrow a \mid o \in L^{opt}, ctx \in L_R, a \in \mathsf{Act}\}$$

For example, the rule $role(bidder) \ : \ \mathsf{goal}(bid(RFT, Bid)) \wedge \mathsf{bel}(rftPublished(RFT)) \rightarrow \mathsf{enact}(bidder)$ is applicable when the OPG-phase has generated the option to enact the *bidder* role, and the agent wants to bid on a published RFT (i.e., *bid*(*RFT*,*Bid*) is in the goal base and *rftPublished*(*RFT*) is in the belief base).

Finally, following [34], we define the capabilities of an agent the states the agent can achieve.

Definition 10 (Capability) The set of capabilities for an agent α is defined as follows:

$$\mathsf{cap}(\alpha) = \{\phi \mid ableToAchieve(\alpha, \phi)\},$$

where $ableToAchieve(\alpha, \phi)$ means that the agent is able to achieve ϕ.

What "able to achieve" means depends on the cognitive agent. For example, in [39] it is suggested that the capability for a GOAL agent to achieve a goal ϕ translates to the existence of goal conditions in action rules and context conditions of modules. In BDI systems with a plan library that associates plans with a triggering event, an agent has the capability to achieve a goal ϕ if it has at least one plan with a triggering event of type *achieve goal* ϕ.

We are now able to define the agents of AORTA:

Definition 11 (AORTA-agent) An AORTA-agent configuration is defined by the following tuple:

$$A = \langle \alpha, \text{MS}, AR, F, C, \mu \rangle,$$

where α is the name of the agent, MS is the mental state, AR is the agents reasoning rules, $AR \subseteq R_A$, F is the set of transition functions (see Definition 14), C is the set of capabilities, $C = \mathsf{cap}(\alpha)$, and $\mu = \langle \mu_{in}, \mu_{out} \rangle$ is the mailbox, contains incoming and outgoing messages.

4.1 Transition system

The semantics allow us to find applicable reasoning rules and use them to update the agent's knowledge bases. We now introduce the transition rules that can be used to make computation runs for AORTA-agents. Each transition rule is only applicable given certain conditions, and by firing a rule, the agent's configuration will change. This makes it possible to compute different traces given an initial state, thus allowing the agents to compute how to best complete their objectives. Our aim is to present a number of transition rules that can be used for organizational reasoning; for which we provide sufficient formal evidence to claim that the rules can be used for organizational reasoning.

Transitions are of the form

$$\frac{\text{premises}}{s \rightarrow s'}$$

where s and s' are agent configurations. If the premises hold, the rule can be executed, which will change the configuration. In the following, we include only the parts of the configuration that are used or updated by the transition rules.

It is the case for all rules below, that they are only applicable if the addition or removal of predicates actually change the configuration. That is, adding a role as an enactment option is only possible, if it is not already an option. To make the rules more easily legible we omit this check in the definitions. The rules can easily be expanded to conform to the constraint following the general patterns, where X refers to a set in the agent's configuration:

$$\frac{\ldots \quad \phi \notin X}{X \rightarrow X \cup \{\phi\}} \qquad \frac{\ldots \quad \phi \in X}{X \rightarrow X \setminus \{\phi\}}$$

Unless otherwise stated, we let α refer to an arbitrary agent's name. This means that, for example, the obligation rules will activate obligations for other role-enacting agents as well as the agent itself. This makes it possible for the agent to reason about obligation fulfillment and violation of other agents, potentially allowing the agents to sanction each other.

4.1.1 Obligation rules

Obligation rules are applicable based on Definition 7, and adds predicates to the organizational knowledge base, enabling agents to use obligations and their violation in reasoning formulas. However, since the rules should apply to the individual agent and not the entire system, we use the metamodel predicates to specify the different phases of the obligation (Figure 1).

When an obligation is activated for a role-enacting agent (i.e. when the condition is fulfilled), the activated obligation is added to the organizational knowledge base.

$$\frac{\mathsf{rea}(\alpha,R) \in \Sigma_o \quad MS \models \mathsf{org}(\mathsf{cond}(R,p,\delta,c)) \wedge \mathsf{bel}(c) \wedge \neg\mathsf{bel}(p)}{\Sigma_o \rightarrow \Sigma_o \cup \{\mathsf{obl}(\alpha,R,p,\delta)\}} \quad \text{(Obl-Activated)}$$

Fulfillment of an obligation simply removes the obligation from the organizational state. Notice, however, that the requirement for fulfillment is only that p follows from the mental state; the deadline may have been reached, but the obligation will still be removed from the knowledge base. We do this to make it easier for agents to distinguish between active and fulfilled obligations.

$$\frac{\mathsf{obl}(\alpha,R,p,\delta) \in \Sigma_o \quad MS \models \mathsf{bel}(p)}{\Sigma_o \rightarrow \Sigma_o \setminus \{\mathsf{obl}(\alpha,R,p,\delta)\}} \quad \text{(Obl-Satisfied)}$$

Violation of an obligation adds the violation to the organizational knowledge base. No other rules adds or removes violations from Σ_o, which means that once an obligation has been violated, the agent(s) can always reason about this (and sanction accordingly).

$$\frac{\mathsf{obl}(\alpha,R,p,\delta) \in \Sigma_o \quad MS \models \neg\mathsf{bel}(p) \wedge \mathsf{bel}(\delta)}{\Sigma_o \rightarrow \Sigma_o \cup \{\mathsf{viol}(\alpha,R,p)\}} \quad \text{(Obl-Violated)}$$

We abbreviate the execution of obligation rules into a single rule, *Obl*. We let *Rule** denote that *Rule* is executed until the configuration no longer changes, and we let ; (semicolon) denote *sequence*, that is, execute the rules delimited by a semicolon in a sequence.

Definition 12 (Obligation execution)

$$\mathit{Obl} \ ::= \ \mathit{Obl\text{-}Activated}^*; \mathit{Obl\text{-}Violated}^*; \mathit{Obl\text{-}Satisfied}^*$$

Note that the execution of each obligation rule is conditional to the instantiation of the premises of each transition. That is, the subsequent executions of rule *Obl-Activated* is applicable when the previous execution of the rule led to a changed configuration. Furthermore, the execution of *Obl-Violated* relies on the condition that the previous execution of *Obl-Activated* left the configuration unchanged.

4.1.2 Option rules

The option rules correspond to the organizational aspects listed in Section 3. Here, we let α denote the agent executing the rules.

We let the agent consider a role, if it is capable of achieving one or more main objectives of the role.

$$\frac{\mathsf{role}(R,Os) \in \Sigma_o \quad \mathsf{rea}(\alpha,R) \notin \Sigma_o \quad \mathsf{cap}(\alpha) \cap Os \neq \emptyset}{\Gamma_o \to \Gamma_o \cup \{\mathsf{role}(R)\}} \quad \text{(Enact)}$$

Note that execution of this rule does not mean that the agent will enact any role it can fulfill, but rather that the agent can consider them as potential roles to enact.

If all main objectives of an enacting role have been completed, the agent should consider whether deacting that role.

$$\frac{\mathsf{role}(R,Os) \in \Sigma_o \quad \mathsf{rea}(\alpha,R) \in \Sigma_o \quad Os \subseteq \Sigma_a}{\Gamma_o \to \Gamma_o \cup \{\neg\mathsf{role}(R)\}} \quad \text{(Deact)}$$

Active obligations concerning an objective should be considered as options. Only the objective of the obligation is part of the option, but the agent can still reason about the deadline and role using reasoning formulas in the context of action rules.

$$\frac{\mathsf{obl}(\alpha,R,p,\delta) \in \Sigma_o \quad \mathsf{obj}(p,SubObj) \in \Sigma_o}{\Gamma_o \to \Gamma_o \cup \{\mathsf{obj}(p)\}} \quad \text{(Objective)}$$

The final option rules are concerned with dependency relations. An agent should consider delegating (or requesting the completion of) an objective (by sending an *achieve*-message), if it is in a dependency relation that enables it to do so. Note that since AORTA only considers the beliefs of each agent in isolation, acting upon a dependency relation can lead to inconsistencies in the agents' behavior. For example, if an agent believes it has delegated a task to another agent, but the second does not hold the same belief.

$$\frac{\{\mathsf{dep}(R_1,R_2,o),\mathsf{rea}(\alpha,R_1)\} \subseteq \Sigma_o \quad \mathsf{obj}(o) \in \Gamma_o}{\Gamma_o \to \Gamma_o \cup \{\mathsf{send}(R_2,achieve,o)\}} \quad \text{(Delegate)}$$

The agent should consider informing about the completion of an objective (by sending a *tell*-message), if other agents depend on that objective.

$$\frac{\{\mathsf{dep}(R_1,R_2,o),\mathsf{rea}(\alpha,R_2)\} \subseteq \Sigma_o \quad MS \models o}{\Gamma_o \to \Gamma_o \cup \{\mathsf{send}(R_1,tell,o)\}} \quad \text{(Inform)}$$

We abbreviate the execution of option rules into a single rule, Opt, which executes each option rule until none of them are applicable.

Definition 13 (Option execution)

$$Opt ::= Enact^*; Deact^*; Objective^*; Delegate^*; Inform^*$$

4.1.3 Action rules

The action transition rules are applicable if there is an option for which the context is entailed by the mental state and the action is defined by the transition function. We distinguish between executing actions and sending a message.

Definition 14 (Action transition function) The action transition function, $\mathscr{A}$, is defined as a partial function $\mathscr{A} : (\mathsf{Act} \times \mathbf{MS}) \to \mathbf{MS}$.

In the following, α is the agent's name.

$$\begin{aligned}
\mathscr{A}(\mathsf{enact}(\rho), MS) &= \langle \Sigma_a, \Gamma_a, \Sigma_o \cup \{\mathsf{rea}(\alpha,\rho)\}, \Gamma_o \cup \{\mathsf{send}(\top, \mathit{tell}, \mathsf{rea}(\alpha,\rho))\}\rangle \\
&\quad \text{if } \Sigma_o \models \exists O.\mathsf{role}(\rho, O) \\
&\quad \text{and } \Sigma_o \not\models \mathsf{rea}(\alpha,\rho) \\
\mathscr{A}(\mathsf{deact}(\rho), MS) &= \langle \Sigma_a, \Gamma_a, \Sigma_o \setminus \{\mathsf{rea}(\alpha,\rho)\}, \Gamma_o \cup \{\mathsf{send}(\top, \mathit{tell}, \neg\mathsf{rea}(\alpha,\rho))\}\rangle \\
&\quad \text{if } \Sigma_o \models \mathsf{rea}(\alpha,\rho) \\
\mathscr{A}(\mathsf{commit}(\phi), MS) &= \langle \Sigma_a, \Gamma_a \cup \{\phi\}, \Sigma_o, \Gamma_o \cup \{\mathsf{send}(\top, \mathit{tell}, \mathsf{obj}(o))\}\rangle \\
&\quad \text{if } \Sigma_a \not\models \phi \text{ and } \Gamma_a \not\models \phi \\
\mathscr{A}(\mathsf{drop}(\phi), MS) &= \langle \Sigma_a, \Gamma_a \setminus \{\phi\}, \Sigma_o, \Gamma_o \cup \{\mathsf{send}(\top, \mathit{tell}, \neg\mathsf{obj}(o))\}\rangle \\
&\quad \text{if } \Gamma_a \models \phi
\end{aligned}$$

Notice that the execution of an action makes changes to two knowledge bases; it generates an option to inform others about the result. We use $\top$ to denote that the message does not have an intended recipient role; it is up to the agent to decide who to inform, if it deems this necessary.

The action execution rule uses the action transition function to change the mental state, if the context for an option holds. Given that the transition function is partial, the rule is only applicable if the transition function is defined.

$$\frac{o \,:\, ctx \to a \in R_A \quad o \in \Gamma_o \quad MS \models ctx \quad \mathscr{A}(a, MS) = MS'}{MS \to MS'} \qquad \text{(Act-Exec)}$$

We assume that the action succeeds once it is acted upon.

Sending a message is possible when the context holds; the agent adds the message to its outgoing mailbox.

$$\frac{o \,:\, ctx \to \mathsf{send}(rcp, msg) \in R_A \quad o \in \Gamma_o \quad MS \models ctx}{\mu_{out} \to \mu_{out} \cup \{\mathsf{msg}(rcp, msg)\}} \qquad \text{(Act-Send)}$$

If no other action is available, the agent has a special *no-op* action available, which does not change the system state. This action is denoted *No-Op*.

We let action execution, *Act*, be either the execution of an action or sending a message. We denote choice by | (vertical bar), meaning that one of the rules should be (nondeterministically) chosen for execution.

Definition 15 (Action execution)

$$\mathit{Act} \;::=\; (\mathit{Act\text{-}Exec} | \mathit{Act\text{-}Send} | \mathit{No\text{-}Op})$$

4.1.4 Other rules

Finally, we define rules for handling external changes and for checking for incoming messages. We assume that the agents share a platform or environment that enables message passing.

We deal with external changes using the following rule:

$$\frac{}{MS \to MS'} \quad \text{(Ext)}$$

Our aim is to define the interaction between the agent and the organization in terms of an organizational reasoning component. The interaction between AORTA, different cognitive agents and the environment is thus not in the scope of this paper, thus we abstract away from dealing with the environment. For a deeper discussion of how to deal with these issues, we refer to [36].

Incoming messages are handled by processing the message through a message transition function, which adds the message to the appropriate knowledge base.

$$\frac{\mathsf{msg}(sender, msg) \in \mu_{in} \quad \mathscr{M}(sender, msg, MS) = MS'}{\begin{array}{c}\mu_{in} \to \mu_{in} \setminus \{\mathsf{msg}(sender, msg)\} \\ MS \to MS'\end{array}} \quad \text{(Check)}$$

Where $\mathscr{M}$ is a message transition function. A simple function will simply put the message into the knowledge base. A more sophisticated function might take into account the sender of the message to, for example, consider whether the sender is trustworthy or if the message should be discarded.

The execution of an entire organizational cycle, *Org*, will then check for messages and external changes, apply obligation rules, generate options and execute an action.

Definition 16 (Organizational cycle execution)

$$Org \ ::= \ Check^*; Ext; Obl; Opt; Act$$

The semantics of AORTA-agents is then defined as computation runs using the transition system.

Definition 17 (AORTA-agent semantics) The semantics of an AORTA-agent is the set of all computation runs for the agent. A computation run is a sequence, $s_0, \ldots, s_n$ or $s_0, \ldots,$, such that each s_i is a configuration, s_0 is the initial configuration, and for all $s_i, i > 0$, we have that a transition $s_i \longrightarrow s_{i+1}$ can be made in the transition system. For finite computation runs, $s_0, \ldots, s_n$, we have that for s_n there is no s_{n+1} such that $s_n \longrightarrow s_{n+1}$.

We write $\rho_x \mapsto \rho_y$ to denote a computation run from a state described by the reasoning formula ρ_x to a state described by the reasoning formula ρ_y, where $\rho_x, \rho_y \in L_R$. Furthermore, we may write $\rho_a \mapsto \rho_b$ where a, b are agents to describe a computation run involving several agents where the state of one agent eventually leads to a state for the other.

5 Evaluation of the AORTA Framework

In this section, we proceed to show how cognitive agents using the AORTA framework are able to reason about organizational matters. We use the RFT scenario to illustrate how the organization-aware agents are able to handle different situations. Since the performance of an AORTA agent depends on the behavior of the cognitive agent, it is difficult to provide general results of a MAS using AORTA without assumptions about the agents. For example, AORTA does not guarantee that an agent will actually achieve the objectives of an organization, since this depends on the agent's own goals and motives, and the organization's ability to sanction violations. Instead, AORTA provides options for the agent to consider, in order

to decide how to act in relation to the organization. The evaluation thus depends on (1) the behavior of the agents, which determines their motivation to participate in the organization, (2) the AORTA reasoning rules that specify how to react to the organizational options and (3) environment dynamics, e.g. time passing. Furthermore, we are not aiming at a full evaluation of the AORTA model. This section aims to show that we have successfully defined a reasoning component that provides agents with organizational reasoning capabilities that are useful in business scenarios such as this one.

5.1 The "request for tender" scenario

In this section we present the scenario used for evaluating AORTA. We consider the *request for tender* (RFT) process, which is a formal, structured invitation to suppliers, to bid, to supply products or services. For example, a company or government may put a building project "out to tender"; that is, publish an invitation for other parties to make a proposal for the building's construction. The aim of the process is to ensure best supplier possible for the requested service or product, such that no parties having the unfair advantage of separate, prior, closed-door negotiations for the contract. The stakeholders of the scenario includes the contracting authority, bidders consortium (possibly consisting of several partners), evaluators, and a publication body.

A system should be able to handle more than one RFT at the same time: i.e. there can be more than one contracting authority putting out an RFT to be fulfilled by different bidder consortia, and possibly advertised by different publication bodies.

The RFT process consists of (at least) the following stages:

1. *Tender elaboration:* decide on terms, conditions, deadlines, etc. for the RFT.
2. *Publication:* publication of the tender and/or distributed to potential bidders.
3. *Request for information:* interested bidders can ask for further information to clarify any uncertainties.
4. *Bid preparation* and (optionally) *consortium formation*.
5. *Bid submission*.
6. *Bid evaluation and decision:* An evaluation team will go through the tenders and decide who will get the contract. Each tender will be checked for compliance and if compliant, then evaluated against the criteria specified in the tender documentation. The tender that offers best value for money will win the business.
7. *Notification:* When a contract has been awarded, the successful tenderer will be advised in writing of the outcome. Unsuccessful tenderers are also informed.
8. *Contract formation:* a formal agreement will be required between the successful tenderer and the contracting authority.

Furthermore, a number of norms can be defined for the scenario:

- Bids must be submitted before the deadline.
- Evaluators have to submit their evaluation on time.
- All bids must be written in English.
- Bids include at least X and at most Y partners.
- Each tender must receive at least Z different bids.
- Evaluators and contractors cannot participate in any bid consortium.
- Bids must be blind.

The scenario is interesting to consider in the context of organizations. It is based on an existing process and is as such well-defined. It is a simple scenario, yet it shows different

kinds of organizational aspects, e.g. explicit definition of roles and objectives. It describes a very strict business process in which the order of certain phases is important. A success criteria of an implemented system is the comparison with evaluation of real-life RFTs. Since it is formally described and based on a number of well-defined stages, organization-aware agents can reason about the best decisions to make in each of the stages. A successful system of organization-aware agents for the RFT process should be able to, e.g., publish different kinds of RFTs that capable agents can bid on, and it should be possible for the agents to choose to violate some of the norms, while taking possible sanctions into account.

5.2 Evaluation

We focus on the part of the RFT process that has to do with bid submission and evaluation. The organization is listed in Table 1 using the metamodel. Notice that the conditional obligations serve two purposes: first, they can be used to describe the obligation to achieve some task (as usual), and second, they can help the agents in understanding the order of execution of the tasks in the organization. For example, the first cond-rule states that a bid must be submitted, before the evaluator can decide if the bid complies with the rules.

We focus on an agent enacting the *evaluator* role and show how AORTA helps the agent adhere to the norms of the system and as efficiently as possible fulfills its responsibilities as an evaluator. The variable binding that takes place when the agents execute their reasoning rules is straightforward and will not be explained in detail. We note, however, that due to the variable binding, the agents will only commit to the RFTs they are interested in.

We initially assume that an agent called *Eve* has generated the options based on her capabilities to enact both the bidder role and the evaluator role using the (Enact) rule. She decides to enact the evaluator role, because she has a goal to evaluate bids for an RFT:

$$\mathsf{role}(evaluator) : \mathsf{goal}(evaluated(Bid, Result)) \rightarrow \mathsf{enact}(evaluator).$$
$$\mathsf{role}(bidder) : \mathsf{goal}(submitted(RFT, Bidder, Bid)) \rightarrow \mathsf{enact}(bidder).$$

Enactment of a role generates the option to tell other agents in the system about this enactment: $\mathsf{send}(\top, tell, \mathsf{org}(\mathsf{rea}(eve, evaluator)))$. She can choose to act upon this option using an action rule, thus allowing other agents to know about her enactment.

All of the agents in the system are shown in Figure 3 and the AORTA action rules are listed in Table 2. Rule (AR1) deals with submission of bids, and (AR2) lets agents inform each other about beliefs. Rules (AR3) and (AR4) are specific to the evaluation: the first rule lets the evaluator commit to perform compliance check of a bid if the bidder did not violate its obligations, and the second rule lets the agent commit to evaluating a bid once a compliance check has been performed.

Since we look at only a part of the system, we assume that the rest of the agents have also enacted roles based on their capabilities, so we proceed to show how they use AORTA for coordination and objective completion. We assume that each agent knows the other agents and which role they enact. We furthermore assume that all agents hold the following beliefs:

- *rft*(*bridge*)
- *published*(*bridge*, *dave*)
- *deadline*(*bridge*, 5, 10, 15)
- *time*(1),

Table 1 Metamodel for the evaluation part of the RFT scenario.

$\mathsf{role}(bidder, \{submitted(RFT, Bidder, Bid)\})$
$\mathsf{role}(evaluator, \{bestTender(RFT, Bid)\})$

$\mathsf{obj}(submitted(RFT, Bidder, Bid), \{\})$
$\mathsf{obj}(bestTender(RFT, Bid), \{complies(RFT, Bid), evaluated(Bid, Result)\})$
$\mathsf{obj}(complies(RFT, Bid), \{\})$
$\mathsf{obj}(evaluated(Bid, Result), \{\})$

$\mathsf{dep}(contractor, bidder, submitted(RFT, Bidder, Bid))$
$\mathsf{dep}(contractor, evaluator, bestTender(RFT, Bid))$
$\mathsf{dep}(evaluator, contractor, submitted(RFT, Bidder, Bid))$

$\mathsf{cond}(bidder, submitted(RFT, Bidder, Bid), complies(RFT, Bid), bid(RFT, Bid))$
$\mathsf{cond}(bidder, submitted(RFT, Bidder, Bid), time(DB),$
$\quad deadline(RFT, DB, DE, DD) \wedge published(RFT, Contractor))$
$\mathsf{cond}(evaluator, complies(RFT, Bid), evaluated(Bid, Result),$
$\quad published(RFT, Contractor) \wedge submitted(RFT, Bidder, Bid) \wedge evaluator(RFT, Evaluator))$
$\mathsf{cond}(evaluator, evaluated(Bid, Result), bestTender(RFT, Bid), complies(RFT, Bid))$
$\mathsf{cond}(evaluator, bestTender(RFT, Bid), biddersInformed(RFT), evaluated(Bid, Result))$
$\mathsf{cond}(evaluator, evaluated(Bid, Result), time(DE),$
$\quad deadline(RFT, DB, DE, DD) \wedge submitted(RFT, Bidder, Bid))$

	Alice	**Bob**	**Charlie**
org	rea(*alice*, *bidder*)	rea(*bob*, *bidder*)	rea(*charlie*, *bidder*)
bel	*bid*(*bridge*, *aliceCo*) *bidDetails*(6)	*bid*(*bridge*, *bobsFirm*) *bidDetails*(10)	*bid*(*bridge*, *charlieAndSon*) *bidDetails*(12)
goal			

	Dave	**Eve**
org	rea(*dave*, *contractor*)	rea(*eve*, *evaluator*)
bel	*evaluator*(*bridge*, *eve*)	*evaluator*(*bridge*, *eve*)
goal	*contractSigned*(C)	*bestTender*(*bridge*, *Bid*)

Fig. 3 The agents being evaluated. Each agent has (from the top) organization beliefs, ordinary beliefs and goals.

Table 2 Action rules available to each of the agents in the system. The predicate *me*(*Name*) identifies the agent by its name.

(AR1) $\mathsf{obj}(submitted(RFT, Bidder, Details)) : \mathsf{bel}(me(Me)) \wedge \mathsf{bel}(bidDetails(X)) \wedge$
$\quad \mathsf{bel}(bid(RFT, Bid)) \rightarrow \mathsf{commit}(submitted(RFT, Me, bid(Bid, X)))$

(AR2) $\mathsf{send}(R, tell, X) : \mathsf{org}(\mathsf{rea}(A, R)) \rightarrow \mathsf{send}(A, \mathsf{bel}(X))$

(AR3) $\mathsf{obj}(complies(RFT, Bid)) :$
$\quad \mathsf{bel}(me(Me)) \wedge \mathsf{bel}(evaluator(RFT, Me)) \wedge \mathsf{bel}(submitted(RFT, Bidder, Details)) \wedge$
$\quad \neg\mathsf{org}(\mathsf{viol}(Bidder, bidder, submitted(RFT, Bidder, Details)))$
$\quad \rightarrow \mathsf{commit}(complies(RFT, Bid))$

(AR4) $\mathsf{obj}(evaluated(Bid, Result)) : \mathsf{bel}(complies(RFT, Bid)) \rightarrow \mathsf{commit}(evaluated(Bid, Result))$

that is, there is a published RFT for a bridge, requested by *Dave*. The deadline predicate, *deadline*(*bridge*, 5, 10, 15), specifies the deadline for bidding (5), evaluation (10) and decision (15), respectively. The time is perceived by the agents and is updated in AORTA using the (Ext) transition rule.

In the following, since we are considering multiple agents, we write bel_α to denote a belief held by agent α. We write bel_* to denote a belief held by every agent in the system

(typically a percept visible to all agents). We use similar conventions for organizational beliefs, organizational options and goals.

We show two things: first, that AORTA lets the agents playing the contractor and evaluator roles decide on the best tender for building a bridge, and second, that the evaluator is able to handle bidders violating the submission deadline by not evaluating them:

$$\mathsf{goal}_{eve}(\mathit{bestTender}(\mathit{bridge}, \mathit{Bid})) \mapsto \mathsf{bel}_{dave}(\mathit{bestTender}(\mathit{bridge}, \mathit{bobsFirm})) \tag{1}$$

$$\mathsf{org}_*(\mathsf{viol}(\mathit{Bidder}, \mathit{bidder}, \mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{bid}(\mathit{BidCompany}, 6)))) \mapsto \neg\mathsf{bel}_{eve}(\mathit{evaluated}(\mathit{bid}(\mathit{BidCompany}, 6), \mathit{Result})) \wedge \neg\mathsf{bel}_{dave}(\mathit{bestTender}(\mathit{bridge}, \mathit{BidCompany})) \tag{2}$$

The first property states that when *Eve* has a goal to find the best tender for the RFT, this will eventually lead to the contractor (in this case, *Dave*) knowing that the best bid came from *Bob*. The second property states that if a bidder violates the submission deadline, their bid will not be evaluated and will not be chosen as the best tender. Note that the properties describe the system from the outside; *Eve* does not initially know that *Bob* is the best tender, but since we designed the system, we expect that she will eventually know it and therefore inform *Dave*.

Initially, the submission-obligations for each of the bidders will be activated. The activation condition for both obligations holds, so both obligations are activated. For the purpose of the evaluation, we focus on the second obligation dealing with the submission deadline defined by the contractor.

(Obl-Activated)
$\mathsf{org}_*(\mathsf{obl}(\mathit{alice}, \mathit{bidder}, \mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details}), \mathit{time}(5)))$
$\mathsf{org}_*(\mathsf{obl}(\mathit{bob}, \mathit{bidder}, \mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details}), \mathit{time}(5)))$
$\mathsf{org}_*(\mathsf{obl}(\mathit{charlie}, \mathit{bidder}, \mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details}), \mathit{time}(5)))$

Notice that the obligations are part of all of the agents' organizational belief base, since all of them know about the agents' role enactment and can reason about the conditional obligation.

Since the obligations concern organizational objectives, options for completing these objectives are generated:

(Objective)
$\mathsf{opt}_{alice}(\mathsf{obj}(\mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details})))$
$\mathsf{opt}_{bob}(\mathsf{obj}(\mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details})))$
$\mathsf{opt}_{charlie}(\mathsf{obj}(\mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details})))$

Finally, using action rule (AR1), each of the agents commit to completing the objective:

(Act-Exec)
$\mathsf{goal}_{alice}(\mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details}))$
$\mathsf{goal}_{bob}(\mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details}))$
$\mathsf{goal}_{charlie}(\mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details}))$

At this point, no reasoning rules are applicable, so the AORTA organizational cycle does not perform any changes to the agent's mental state. It is thus up to the agents (using their capabilities) to complete the objectives they have committed to achieving. Then, one of two things happen: either the deadline for submission is reached and the obligation to submit is violated, or the bids are submitted and the obligation is satisfied.

We assume that all of the bidders have the capability to submit their bid, but to make things interesting, we let *Alice* submit her bid after the deadline has been reached to show how the contractor and the evaluator can react to this violation using AORTA.

After a while, *Bob* and *Charlie* submit their bids:

(Ext) $\mathsf{bel}_*(\mathit{time}(4))$
$\mathsf{bel}_{\mathit{bob}}(\mathit{submitted}(\mathit{bridge}, \mathit{bob}, \mathit{bid}(\mathit{bobsFirm}, 10)))$
$\mathsf{bel}_{\mathit{charlie}}(\mathit{submitted}(\mathit{bridge}, \mathit{charlie}, \mathit{bid}(\mathit{charlieAndSon}, 12)))$

The dependency relation between contractor and bidder regarding submission generates an option to inform contractors about the submission. The agents use (AR2) to inform the contractor:

(Inform) $\mathsf{opt}_{\mathit{bob}}(\mathsf{send}(\mathit{contractor}, \mathit{tell}, \mathit{submitted}(\mathit{bridge}, \mathit{bob}, \mathit{bid}(\mathit{bobsFirm}, 10))))$
$\mathsf{opt}_{\mathit{charlie}}(\mathsf{send}(\mathit{contractor}, \mathit{tell}, \mathit{submitted}(\mathit{bridge}, \mathit{charlie}, \mathit{bid}(\mathit{charlieAndSon}, 12))))$

(Act-Send) $\mathsf{msg}(\mathit{dave}, \mathsf{bel}(\mathit{submitted}(\mathit{bridge}, \mathit{bob}, \mathit{bid}(\mathit{bobsFirm}, 10)))) \in \mu_{out}^{bob}$
$\mathsf{msg}(\mathit{dave}, \mathsf{bel}(\mathit{submitted}(\mathit{bridge}, \mathit{charlie}, \mathit{bid}(\mathit{charlieAndSon}, 12)))) \in \mu_{out}^{charlie}$

The contractor receives the bids and adds them to the belief base:

(Check) $\mathsf{bel}_{\mathit{dave}}(\mathit{submitted}(\mathit{bridge}, \mathit{bob}, \mathit{bid}(\mathit{bobsFirm}, 10)))$
$\mathsf{bel}_{\mathit{dave}}(\mathit{submitted}(\mathit{bridge}, \mathit{charlie}, \mathit{bid}(\mathit{charlieAndSon}, 12)))$

We note that *Alice* has not yet submitted her bid. Let us assume we reach the submission deadline, leading to the violation of her submission obligation:

(Ext) $\mathsf{bel}_*(\mathit{time}(5))$

(Obl-Violated) $\mathsf{org}_*(\mathit{viol}(\mathit{alice}, \mathit{bidder}, \mathit{submitted}(\mathit{bridge}, \mathit{Bidder}, \mathit{Details})))$

Similarly to the activation of obligations, the violation of *Alice*'s obligation is part of every agents' organizational belief base. This means that, at this point, all agents in the system know that *Alice* has violated her obligation.

We assume that she eventually submits a bid, which is received by the contractor:

(Check) $\mathsf{bel}_{\mathit{dave}}(\mathit{submitted}(\mathit{bridge}, \mathit{alice}, \mathit{bid}(\mathit{aliceCo}, 6)))$

To summarize, the bidders have used the organizational model and AORTA to (1) commit to submitting their bids and (2) inform the contractor about this. We now turn to the evaluator role, which depends on the contractor for receiving the bids that should be evaluated. Based on this dependency relation, the contractor generates options using the (Inform) rule to inform the evaluator about the bids and acts upon these options by sending the submissions to *Eve*:

(Check) $\mathsf{bel}_{\mathit{eve}}(\mathit{submitted}(\mathit{bridge}, \mathit{alice}, \mathit{bid}(\mathit{aliceCo}, 6)))$
$\mathsf{bel}_{\mathit{eve}}(\mathit{submitted}(\mathit{bridge}, \mathit{bob}, \mathit{bid}(\mathit{bobsFirm}, 10)))$
$\mathsf{bel}_{\mathit{eve}}(\mathit{submitted}(\mathit{bridge}, \mathit{charlie}, \mathit{bid}(\mathit{charlieAndSon}, 12)))$

The submissions activate obligations for the evaluator to evaluate the bids before the evaluation deadline is reached. Furthermore, the evaluator is obliged to perform compliance check

for each of the submissions before evaluating them:

(Obl-Activated) $\mathsf{org}_*(\mathsf{obl}(eve, evaluator, evaluated(bid(aliceCo, 6), \text{Result}), time(10))$
$\mathsf{org}_*(\mathsf{obl}(eve, evaluator, complies(bridge, bid(aliceCo, 6)), evaluated(bid(aliceCo, 6), \text{Result})))$
...

(Objective) $\mathsf{opt}_{eve}(\mathsf{obj}(evaluated(bid(aliceCo, 6), \text{Result})))$
$\mathsf{opt}_{eve}(\mathsf{obj}(complies(bridge, bid(aliceCo, 6))))$
...

The evaluator can now consider the generated objective options using its reasoning rules (AR3) and (AR4). Since we assume that she has the capabilities required for her role, *Eve* will eventually complete the objectives she commits to. Rule (AR3) concerns the compliance check and is only applicable if the submission deadline for a given bid was not violated. Rule (AR4) concerns evaluation and is only applicable if the evaluator has verified that the submission complies to the terms and conditions of the RFT. We thus assume that eventually, the bids from *Bob* and *Charlie* will be evaluated. Furthermore, since she already has adopted the goal to find the best tender, she will eventually make that decision:

(Ext) $\mathsf{bel}_{eve}(evaluated(bid(bobsFirm, 10), won))$
$\mathsf{bel}_{eve}(evaluated(bid(charlieAndSon, 12), lost))$
$\mathsf{bel}_{eve}(bestTender(bridge, bobsFirm))$

We should note that while the action rules of AORTA makes *Eve* skip the evaluation of one of the bidders, she may still choose to do so if it is in her own interest. However, since the contractor can also reason about the violation, he may still choose to ignore the bid from *Alice*. The dependency relation between contractor and evaluator generates an option to inform the contractor about the best tender, which the evaluator then acts upon using (AR2):

(Inform) $\mathsf{opt}_{eve}(\mathsf{send}(contractor, tell, bestTender(bridge, bobsFirm)))$

(Act-Send) $\mathsf{msg}(dave, \mathsf{bel}(bestTender(bridge, bobsFirm))) \in \mu^e_{out}$

(Check) $\mathsf{bel}_{dave}(bestTender(bridge, bobsFirm))$

This concludes the execution of the RFT scenario. We have shown that the bids have been evaluated by the evaluator and the best tender was chosen, after which this information was passed on to the contractor.

We have thus shown that by using AORTA the organizational model in the system is made available for the agents to act upon. In the OPG-phase, the framework generates options in relation to the model and the state of the system, helping the agents decide what to do in relation to the organization. Since the capabilities of the agents are preexistent, AORTA mainly provides a functional way for the agents to use their capabilities in completing the organizational objectives. Furthermore, in the OC-phase, obligations are activated for the agents, and are available to not only responsible agents, but other agents as well, giving a way for them to react to fulfillment and violation by other agents in the system. We showed that the evaluator could decide not to evaluate a submission if the was submitted after the submission deadline. This was possible exactly because the OC-phase handles obligations for every agent, not just the agent itself.

How the agents use AORTA is specified by the action rules, and they should be specified by the programmer. While very general rules can be established (e.g. always enact possible

roles, always commit to objectives), more specific rules allow the agents to better react to preconditions for performing a task or to possible violations.

The fact that each agent uses a separate AORTA component means that the agents may hold different beliefs about the organization, just as they may have different views of their environment. The (Inform) rule and the enact-action accommodate this by generating send-options for maintaining consistency among the agents. For example, when *Eve* enacts the evaluator role, an option is generated to tell other agents in the system about this, and when an agent enacting the bidder role submits an application, an option is generated to inform the contractor about this. The second option can then be acted upon using (AR2), which uses the agents' beliefs about role enactment to tell the contractor about the submission.

Furthermore, since the component is added to the agent and assumes nothing a priori about the agent, the plans inside the agent may very well conflict with the AORTA action rules. For example, the agent may choose not to commit to an objective using AORTA, but adopt it anyway using its internal plans. However, if this leads to a violation of an organizational obligation, other agents will be aware of this, and can choose to punish the agent. Thus, AORTA allows the agents to follow the rules of an organization, but poses no restrictions on them to do so; if they choose to follow their own plans, this cannot be prevented – and rightfully so: otherwise their autonomy would be severely limited.

6 Conclusion

In this paper, we present AORTA, a framework to provide agents with means to reason about, enter and leave an organization in open environments in a way that does not take away all of their autonomy. AORTA can be seen as an add-on component to cognitive agents, providing them with organizational reasoning capabilities. These capabilities are *general* and not based on a specific organizational model or agent framework, but is designed to allow different kinds of agents to reason about different kinds of organizational models.

AORTA is used by each agent individually, which first of all means that the agents can decide by themselves how to use the component, making AORTA suitable for many different kinds of agents. It may, however, also lead to agents holding different views about the organization, leading to inconsistency among them. This is not different from what happens in real-life, where actors in an environment may very well have different views about the world state, and is as such not something we believe is particularly problematic.

The framework is founded in formal operational semantics that precisely define how the agent can reason about an organization. Using temporal logic, we have formalized the intended behavior of organization-aware agents and have subsequently captured this behavior using operational semantics. Note that there is currently no formal connection between these semantics, and future work is needed in order to establish the correctness of the operational semantics with respect to the semantics of obligations.

Organizations in AORTA are represented by an organizational metamodel. We have shown that agents can use the metamodel to reason about a system in which that organizational model is in effect. The metamodel supports the notions of roles and role enactment, objectives and objective dependencies, and conditional obligations. Using transition rules, the component can activate obligations, detect violations and suggest role enactment or objective commitments, all based on the agent's current state. The agents act upon this using organizational actions, allowing to following the suggestions – enact roles or commit to objectives – or to coordinate their actions with other agents in the system.

We have provided an initial evaluation of the the framework using a scenario in which a company puts a building project out to tender, i.e. requests other parties to make a proposal for the building's construction. We showed that, given agents with the capabilities to fulfill the roles in such a scenario, AORTA made it possible for them to coordinate their tasks and detect and act upon violations of the obligations imposed upon the agents.

Since AORTA provides organizational reasoning capabilities without taking away the agents' autonomy, the agents can by themselves choose how to make use of the framework. That is, AORTA may generate options for committing to certain objectives or coordinating with other agents, but the agents are free to choose not to act upon these options. Furthermore, even if an AORTA reasoning rule is designed to e.g. punish a violating agent, the agent's own intentions may conflict with these rules; AORTA poses no restrictions on this, since this would limit the agent's autonomy. As argued, even if an agent then violates the organizational obligations, other agents can detect this, and may choose to punish accordingly. If they do not punish, then either the obligations are unnecessary, or the agents are not interested in fulfilling their roles.

As mentioned, we have implemented an earlier version of the framework in Java and have integrated it with the *Jason* platform [30]. We plan to extend that work with the complete and updated operational semantics presented in this paper. This would make it possible to execute and evaluate larger scenarios, and to, e.g., test how agents are able to recover from violations of obligations.

AORTA takes the perspective of the agent and provides it with capabilities to reason in a system with an organization. Therefore, the focus of the framework is on the agent's beliefs about the organization and the environment, meaning that there is no notion of an organizational entity that the agents can interact with. If we instead take the organization's perspective, we are required to look at it from the point of view of an organizational entity; this entity would exist in the environment and would contain the state of the organization. Such entity is called an *artifact* [27] and agents are able to interact with it to e.g. enact roles or form groups. We plan to investigate how to integrate AORTA with organizational artifacts, so that e.g. the enactment action is connected to an organizational artifact.

Appendix: Translating an OperA model

The OperA model [17] proposes an expressive way for defining open organizations distinguishing explicitly between the organizational aims, and the agents who act in it. OperA enables the specification of the organizational structure, requirements and objectives, and at the same time allows participants to have the freedom to act according to their own capabilities and demands. At an abstract level, an OperA model describes the aims and concerns of the organization with respect to the social system. These are described as the organization's externally observable objectives, i.e. the desired states of affairs for the organization.

The OperA model contains the *Organizational Model* (OM), which is the result of the observation and analysis of the domain and describes the desired behavior of the organization, as determined by the organizational stakeholders in terms of objectives, norms, roles, interactions and ontologies. The OM consists of four interrelated *structures*: the social, interaction, normative and communicative structure. The *social structure* of an organization describes the roles holding in the organization. It consists of a list of role definitions, group definitions, and a role dependencies graph. The *interaction structure* describes the states that the agents should achieve, in terms of meaningful scenes that follow pre-defined abstract scene scripts. A scene script describes a scene by its players (roles), its desired results

and the norms regulating the interaction. A scene script establishes also the desired interaction patterns between roles, that is, a desired combination of the (sub-) objectives of the roles. The *normative structure* describes expectations and boundaries for agent behavior, and the *communicative structure* specifies the ontology and the communication language used in the society.

In many ways, the OperA model is richer than the AORTA metamodel. The notion of, e.g., groups and scenes is not present in AORTA, and they cannot be directly translated into an equivalent notion. Furthermore, in some cases, the translation may not be ideal for practical purposes, so it may be necessary to manually change the metamodel after translating the OperA model. Even so, the translation provides a starting point for creating a metamodel, which should be preferred to creating the entire metamodel manually.

In the following we focus on the social, interaction and normative structure, and show how a subset of the OperA model can be translated into the AORTA metamodel. We do not directly consider the communicative structure, but since the ontology defined here is used in the other structures, the communication language is inherently included in the AORTA metamodel as well.

Social structure

The social structure specifies the roles, groups and role dependencies of the organization. Since the metamodel does not contain the notion of groups, we show only how to translate roles and role dependencies. A role in OperA is a tuple

$$role(r, Obj, Sbj, Rgt, Nor, tp),$$

where r is the role identifier, *Obj* and *Sbj* are objectives and sub-objectives, *Rgt* is the set of rights associated with the role, *Nor* is the set of norms and *tp* is the type of the role (which can be either external or institutional).

The metamodel has no concept of rights and does not distinguish between different role types. A role can therefore at most be described by its identifier, objectives, sub-objectives and norms. An OperA role then becomes $\mathsf{role}(r, Obj)$ in the metamodel. We show how role norms can be translated in the normative structure. A set of sub-objectives for objective γ is defined in OperA as a set $\Pi\gamma = \{\gamma_1, \ldots, \gamma_n\}$ such that $\bigwedge_{i=1}^{n} \gamma_i \rightarrow \gamma$. For each objective, γ, we thus add $\mathsf{obj}(\gamma, \Pi\gamma)$ to the metamodel.

The social structure defines three kinds of dependency relations: hierarchical, market and network relations. They differ in how agents in the relation have authority over one another: the market relation facilitates *bidding*, the network relation is based on *requests*, and a hierarchical relation uses *delegation*. In AORTA, a dependency relation between roles r_1 and r_2 means that r_1 depends on r_2 for the completion of an objective. The AORTA dependency relation is not based on one agent having authority over another, thus it is up to the agents to decide how to use it, for example, by delegate tasks to agents enacting r_2. We thus perform the following translation: an OperA relation $r_1 \succeq_\gamma r_2$ becomes $\mathsf{dep}(r_1, r_2, \gamma)$ in the metamodel.

Interaction structure

The interaction structure divides the organizational activity into scene scripts that provide partial ordering of objectives and transitions between scenes that provide synchronization

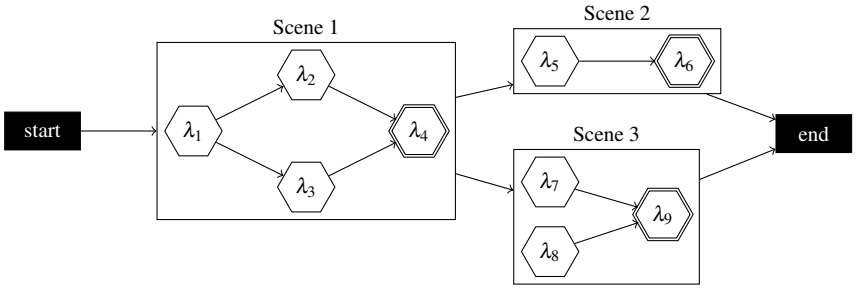


Fig. 4 Interaction structure with landmark patterns for each scene script. Landmarks with a double edge are results of the scene script.

and evolution of roles. Objectives of scenes are partially ordered using landmark patterns, which represent the minimum requirements for achieving the results of the scene.

Figure 4 shows an interaction structure with three scene scripts. We use this to show how to perform the translation into the metamodel. Informally, the interaction structure tells us that scene 1 ought to be completed before beginning on scene 2 and 3, and either scene 2 or scene 3 ought to be completed before moving on to the end scene. Furthermore, looking at scene 1, landmark λ_1 should be completed before λ_2 and λ_3, and so on. Since AORTA does not have a notion of scenes, we propose a translation of the interaction structure to conditional obligations with deadline. We basically convert each landmark pattern to a set of conditional obligations, and we connect the scenes using conditional obligations as well.

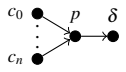
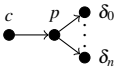
An obligation $O(p < \delta \mid c)$ means that p ought to be achieved once c is achieved, and before δ occurs. We thus propose a translation where each landmark is the objective of an obligation, with the previous landmark(s) as condition and the next landmark(s) as deadline. For example, given a landmark pattern $\lambda_a \leq \lambda_b \leq \lambda_c$, the obligation to achieve λ_b becomes $O(\lambda_b < \lambda_c \mid \lambda_a)$, which states that the agent ought to achieve λ_b once λ_a has been achieved, but before λ_c is achieved, corresponding to the landmark pattern. In other patterns where multiple landmarks must be achieved in parallel before the next landmark, the obligation becomes a bit more complex, since we have to incorporate this. However the well-defined semantics of landmark patterns [32] and of scene transitions make it possible to translate such patterns into conditional obligations that correspond well with the meaning of the pattern. Table 3 lists the different kinds of patterns that can appear in the interaction structure.

We deal with the edge cases as follows. If for a landmark λ there is no landmark λ', such that $\lambda' \leq \lambda$, then the condition for the obligation to achieve λ is $\top$. This corresponds to an obligation that is immediately activated. If for a landmark λ there is no λ', such that $\lambda \leq \lambda'$, then the deadline for the obligation to achieve λ is $\bot$. This makes it hard to detect a violation, since an obligation is only violated once the deadline is reached, which never happens in this case. In order to accommodate this, it will be necessary to manually change the resulting metamodel to incorporate actual deadlines in these cases.

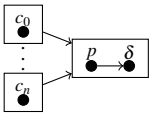
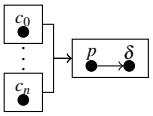
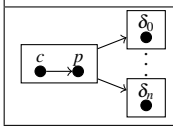
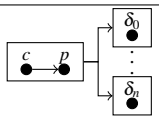
Example 1 We can use the translation scheme above to translate the interaction structure in Figure 4. The first objective, λ_1, is translated into the obligation $O(\lambda_1 < \lambda_2 \vee \lambda_3 \mid \top)$. The condition is $\top$ since λ_1 is the first objective. The deadline is reached when either λ_2 or λ_3 have been achieved, corresponding the fact that they are after λ_1 in the partial ordering. The

Table 3 Different types of landmark patterns and scene transitions and their corresponding conditional obligations with deadline.

Landmark patterns

	$O(p < \delta \mid c_0 \wedge \cdots \wedge c_n)$		$O(p < \delta_0 \vee \cdots \vee \delta_n \mid c)$

Scene transitions

	$O(p < \delta \mid c_0 \vee \cdots \vee c_n)$		$O(p < \delta \mid c_0 \wedge \cdots \wedge c_n)$
	$O(p < \delta_0 \vee \cdots \vee \lambda_n \mid c)$		$O(p < \delta_0 \vee \cdots \vee \lambda_n \mid c)$

rest of the resulting obligations are shown below.

$$O(\lambda_2 < \lambda_4 \mid \lambda_1) \qquad O(\lambda_3 < \lambda_4 \mid \lambda_1)$$
$$O(\lambda_4 < \lambda_5 \vee \lambda_7 \vee \lambda_8 \mid \lambda_2 \wedge \lambda_3) \qquad O(\lambda_5 < \lambda_6 \mid \lambda_4)$$
$$O(\lambda_6 < \bot \mid \lambda_5) \qquad O(\lambda_7 < \lambda_9 \mid \lambda_4)$$
$$O(\lambda_8 < \lambda_9 \mid \lambda_4) \qquad O(\lambda_9 < \bot \mid \lambda_7 \wedge \lambda_8)$$

In this case, the obligations to achieve λ_6 and λ_9 both have $\bot$ as deadline, so it may be necessary to manually change the obligations to use actual deadlines.

The translation shown above omits a few parts of the interaction structure. Transitions between scenes may include *role evolution* relations, specifying how agents can (or are obliged) to enact a role in the next scene based on there current role. Furthermore, a role evolution may specify a *conflict* between roles, i.e. that two roles may not be simultaneously enacted by a single agent. This is not captured in the translation above. A role evolution relation is defined by

$$\textit{role-evolution}(s.r_1, t.r_2, SN, \lambda),$$

where s and t are scenes, r_1 is a role in s, r_2 a role in t, SN is the type of relation and can be either *necessary*, *sufficient* or *conflict*, and λ is the set of conditions for performing the role evolution (the landmarks that must be fulfilled by the agent). As we are considering only obligations, we are not translating sufficient role evolutions (i.e. the agent is allowed to enact a certain role in the next scene).

A necessary role evolution is translated into the following conditional obligation:

$$O_{r_1}(\mathsf{rea}(Ag, r_2) < \lambda_t \mid \lambda),$$

where Ag is the agent, λ_t is the set of initial landmarks of scene t, and λ is the set of conditions for performing the role evolution.

Role conflicts are somewhat problematic in AORTA. Intuitively, a role conflict is translated into the following conditional obligation:

$$O_{r_1}(\neg\mathsf{rea}(Ag, r_2) < \lambda_t^r \mid \lambda),$$

where Ag is the agent, λ_t^r is the set of results of scene t, and λ is the set of conditions for the role evolution. However, if the obligation is activated and Ag is not enacting r_2, the obligation is immediate satisfied, and can therefore never be violated. The issue stems from the fact that obligations in AORTA are based on *achievement*, not *maintenance*, which means we cannot specify an obligation to maintain a state (in this case $\neg\mathsf{rea}(Ag,r_2)$. A work-around is to let the activation condition be based on the enactment of *both* roles:

$$O_{r_1}(\neg\mathsf{rea}(Ag,r_2) < \lambda_t^r \mid \mathsf{rea}(Ag,r_2) \wedge \lambda).$$

That is, whenever the agent enacts both roles, it is obliged to deact the second role. Similarly, a global role conflict can be translated into $O_{r_1}(\neg\mathsf{rea}(Ag,r_2) < \bot \mid \mathsf{rea}(Ag,r_2))$.

Normative structure

The normative structure defines the expected boundaries of the agents participating in an organization. This is done by specifying *norms* that describe what agents enacting specific roles are expected to do. Norms in the normative structure are divided into *role norms*, *scene norms* and *transition norms*. Role norms define the generally expected behavior of a role regardless of participation in scenes. Scene norms define the expected behavior of roles participating in a specific scene. Transition norms define the limitations related with enacting new roles when moving between scenes, and were handled above in the translation of the interaction structure.

Norms in OperA are specified in Logic for Contract Representation (LCR). Different types of obligations are defined: conditional obligations, obligations with deadline and obligations without deadline. A obligation to achieve p before δ when c in LCR is translated to $O(p < \delta \mid c)$ in the metamodel. If $c = \top$, the obligation has no condition. If $\delta = \bot$, the obligation has no deadline, which as discussed causes some issues regarding violation detection.

Example: Request for tender

The RFT organization has been implemented in OperA. Our implementation of the bidding procedure is somewhat simple, in that we assume that the bidders simply specify a price for the *RFT* and the evaluator chooses the lowest bid. Since the process of making such decisions is not in the scope of this paper, we believe this is justified. We briefly describe the implementation and show its translation to the AORTA metamodel. The organization contains five roles, as shown in the social structure in Figure 5. The social structure furthermore defines dependencies between the roles, based on the organization's objectives. The roles are translated into the following roles in the metamodel:

$$\mathsf{role}(\mathit{bidder}, \{\mathit{submitted}(\mathit{RFT},\mathit{Bidder},\mathit{Bid}), \mathit{contractSigned}(\mathit{Contract})\})$$
$$\mathsf{role}(\mathit{contractor}, \{\mathit{rft}(\mathit{RFT}), \mathit{published}(\mathit{RFT},\mathit{Contractor}), \mathit{contractSigned}(\mathit{Contract}), \mathit{bestTender}(\mathit{RFT},\mathit{Bid})\})$$
$$\mathsf{role}(\mathit{evaluator}, \{\mathit{bestTender}(\mathit{RFT},\mathit{Bid})\})$$
$$\mathsf{role}(\mathit{publicationBody}, \{\mathit{published}(\mathit{RFT},\mathit{Contractor})\})$$
$$\mathsf{role}(\mathit{consortiumPartner}, \{\mathit{submitted}(\mathit{RFT},\mathit{Bidder},\mathit{Bid}), \mathit{consortium}(\mathit{Bid},\mathit{Partners})\})$$

Each role is associated with a number of main objectives. Most of these objectives are in turn associated with a number of subobjectives. We will not go into details with the objective

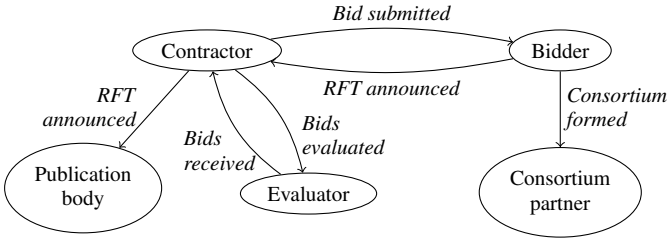


Fig. 5 The social structure of the RFT scenario. Each node corresponds to a role, and each edge corresponds to an objective dependency between roles.

specification, but will use the objectives in the translations below. The objectives are mostly self-explanatory, but will be explained in detail when deemed necessary.

The dependency relations are translated into the following predicates:

$$\mathsf{dep}(bidder, contractor, published(RFT, Contractor))$$
$$\mathsf{dep}(bidder, consortiumPartner, consortium(Bid, Partners))$$
$$\mathsf{dep}(contractor, evaluator, bestTender(RFT, Bid))$$
$$\mathsf{dep}(contractor, publicationBody, published(RFT, Contractor))$$
$$\mathsf{dep}(contractor, bidder, submitted(RFT, Bidder, Bid))$$
$$\mathsf{dep}(evaluator, contractor, submitted(RFT, Bidder, Bid))$$

The dependency relations let the agents know how to cooperate in order to achieve the objectives. For example, a publication body agent can reason that since the contractor depends on it for publishing the RFT, it can request information about the RFT from the contractor, in order to fulfill its objectives. Furthermore, the contractor can ask the publication body to achieve the goal and provide the agent with the information required to publish it. Notice that the contractor depends on bidders for the submission of bids, while the evaluators depend on the contractor for this information.

The RFT process consists of a number of stages, which are translated into scenes in the interaction structure (Figure 6). The contractor should enroll a publication body and a number of evaluators. The publication body will publish the RFT once its terms, conditions and deadlines have been decided by the contractor. Potential bidders can prepare a bid, including requesting more information and forming a consortium. They can then submit their bids (before the deadline). In the evaluation process, the evaluators ensure that the bids are compliant and decide on the best tender. Finally, the contract is then awarded to the best tender, and the parts form and sign the contract.

The following is a subset of the conditional obligations generated from the IS:

$$O_{bidder}(bid(RFT, Bid) < submitted(RFT, Bidder, Bid) \mid consortium(Bid, Partners) \wedge infoRequested(Bid, Request))$$
$$O_{contractor}(rft(RFT) < published(RFT, Contractor) \mid deadline(RFT, DBid, DEvaluation, DDecision) \wedge terms(RFT, Terms) \wedge conditions(RFT, Conditions))$$
$$O_{contractor}(contract(Contract, RFT, Bid) < contractSigned(Contract) \mid biddersInformed(RFT))$$
$$O_{evaluator}(complies(RFT, Bid) < evaluated(Bid, Result) \mid published(RFT, Contractor) \wedge submitted(RFT, Bidder, Bid) \wedge evaluator(RFT, Evaluator))$$

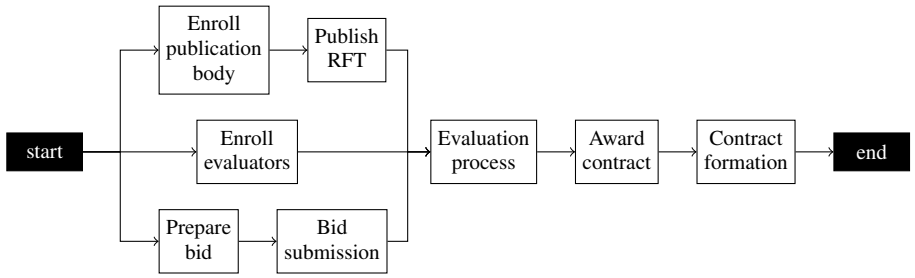


Fig. 6 The interaction structure of the RFT scenario. The forked arcs indicate that all incoming/outgoing scenes must be realized.

For example, the contractor should only finalize the RFT once deadline, terms and conditions have been decided, and it should be done before it is being published. Similarly, the evaluator should ensure that a bid complies with the terms and conditions of an RFT before evaluating it, and should only do so once the RFT has been published, the bids are submitted and the evaluators have been assigned.

Finally, the normative structure includes the norms defined in Section 5.1. Since the translation is rather straightforward, we simply show a few of the obligations as defined in the metamodel:

$$\begin{aligned} &O_{bidder}(submitted(Bid, Bidder, Details) < time(DBid) \\ &\qquad \mid deadline(RFT, DBid, DEvaluation, DDecision) \wedge \\ &\qquad published(RFT, Contractor)) \\ &O_{evaluator}(evaluated(Bid, Result) < time(DEvaluation) \\ &\qquad \mid deadline(RFT, DBid, DEvaluation, DDecision) \wedge \\ &\qquad submitted(RFT, Bidder, Bid)) \end{aligned}$$

The norms allow for more specific obligations concerning the objectives. For example, bid submission should happen before the bidding deadline (*DBid*), and evaluation must be completed before the evaluation deadline (*DEvaluation*).

The norm concerning evaluators and contractors participating in a bidding consortium may be considered a role conflict, which in the metamodel can be expressed as follows:

$$\begin{aligned} &O_{evaluator}(\neg \mathsf{rea}(\alpha, bidder) < evaluated(Bid, Result) \\ &\qquad \mid \mathsf{rea}(\alpha, bidder) \wedge evaluator(RFT, \alpha) \wedge bid(RFT, Bid) \wedge \\ &\qquad consortium(Bid, Partners) \wedge \alpha \in Partners). \end{aligned}$$

That is, an agent enacting the *evaluator* role is obliged to deact the *bidder* role, before the bid is evaluated, *if* that agent is an evaluator in an RFT for which it is also a consortium partner. A similar obligation can be specified for contractors.

References

1. Aldewereld, H., Dignum, V., Jonker, C.M., van Riemsdijk, M.B.: Agreeing on Role Adoption in Open Organisations. Künstliche Intelligenz **26**(1), 37–45 (2011)
2. Alechina, N., Dastani, M., Logan, B.: Programming norm-aware agents. AAMAS 12 Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems **2** (2012)

3. Belnap, N., Perloff, M., Xu, M.: Facing the Future: Agents and Choices in Our Indeterminist World. Oxford University Press (2001)
4. Boissier, O., Riemsdijk, M.B.: Organisational reasoning agents. In: Agreement Technologies, Law, Governance and Technology Series. Springer (2013)
5. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons (2007)
6. Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., van der Torre, L.: The BOID architecture: conflicts between beliefs, obligations, intentions and desires. AAMAS '06 pp. 9–16 (2001)
7. Carabelea, C., Boissier, O., Castelfranchi, C.: Using Social Power to Enable Agents to Reason About Being Part of a Group. In: Engineering Societies in the Agents World V, pp. 166–177 (2005)
8. Castelfranchi, C., Dignum, F., Jonker, C.M., Treur, J.: Deliberate Normative Agents: Principles and Architecture. Intelligent Agents VI **LNAI 1757**, 364–378 (2000)
9. Criado, N., Argente, E., Noriega, P., Botti, V.: Towards a normative BDI architecture for norm compliance. In: COIN@MALLOW2010 (2010)
10. Dastani, M.: 2APL: A Practical Agent Programming Language. Autonomous Agents and Multi-Agent Systems **16**(3), 214–248 (2008). DOI 10.1007/s10458-008-9036-y
11. Dastani, M., Dignum, V., Dignum, F.: Role-assignment in open agent societies. In: AAMAS '03, pp. 489–496 (2003)
12. Dastani, M., van Riemsdijk, M.B., Hulstijn, J., Dignum, F., Meyer, J.J.: Enacting and deacting roles in agent programming. In: Agent-Oriented Software Engineering V, *Lecture Notes in Computer Science*, vol. 3382, pp. 189–204. Springer (2005)
13. Dignum, F., Dignum, V.: A formal semantics for agent (re)organization. Journal of Logic and Computation pp. 61–76 (2013). URL http://logcom.oxfordjournals.org/content/early/2013/11/22/logcom.ext058.short
14. Dignum, F., Dignum, V., Thangarajah, J., Padgham, L., Winikoff, M.: Open agent systems??? In: Agent-Oriented Software Engineering VIII, pp. 73–87. Springer (2008)
15. Dignum, F., Kinny, D., Sonenberg, L.: From desires, obligations and norms to goals. Cognitive Science Quarterly **2**(3-4), 405–427 (2002)
16. Dignum, F., Morley, D., Sonenberg, E.a., Cavedon, L.: Towards socially sophisticated BDI agents. In: Proceedings Fourth International Conference on MultiAgent Systems, pp. 111–118. IEEE Comput. Soc (2000)
17. Dignum, V.: A model for organizational interaction: based on agents, founded in logic. Ph.D. thesis, Utrecht University (2004)
18. Dignum, V., Dignum, F.: What's in it for me? Agent deliberation on taking up social roles. In: EUMAS 2004 (2004). URL http://dspace.library.uu.nl/handle/1874/11493
19. Dignum, V., Dignum, F.: A logic of agent organizations. Logic Journal of the IGPL **20**(1), 283–316 (2011)
20. Esteva, M., de la Cruz, D., Sierra, C.: Islander: An electronic institutions editor. In: AAMAS '02 (2002). DOI 10.1145/545056.545069
21. Esteva, M., Rosell, B., Rodriguez-Aguilar, J.A., Arcos, J.L.: Ameli: An agent-based middleware for electronic institutions. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS '04, pp. 236–243. IEEE Computer Society, Washington, DC, USA (2004). DOI 10.1109/AAMAS.2004.56. URL http://dx.doi.org/10.1109/AAMAS.2004.56
22. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: an organizational view of multi-agent systems. Agent-Oriented Software Engineering IV **LNCS 2935**(July 2003), 214–230 (2004)

23. Grossi, D., Aldewereld, H., Dignum, F.: Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In: P. Noriega, J. Vzquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara, E. Matson (eds.) Coordination, Organizations, Institutions, and Norms in Agent Systems II, *Lecture Notes in Computer Science*, vol. 4386, pp. 101–114. Springer Berlin Heidelberg (2007). DOI 10.1007/978-3-540-74459-7_7. URL http://dx.doi.org/10.1007/978-3-540-74459-7_7
24. Grossi, D., Meyer, J.J.C., Dignum, F.: Counts-as: Classification or constitution? an answer using modal logic. In: L. Goble, J.J. Meyer (eds.) Deontic Logic and Artificial Normative Systems, *Lecture Notes in Computer Science*, vol. 4048, pp. 115–130. Springer Berlin Heidelberg (2006)
25. Hindriks, K.V.: Programming Rational Agents in GOAL. Multi-Agent Programming: Languages, Tools and Applications pp. 119–157 (2009)
26. Hormazbal, N., Cardoso, H., de la Rosa, J.L., Oliveira, E.: An approach for virtual organisations dissolution. In: Coordination, Organizations, Institutions and Norms in Agent Systems V, *Lecture Notes in Computer Science*, vol. 6069, pp. 70–85. Springer (2010)
27. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents. Autonomous Agents and Multi-Agent Systems **20**(3), 369–400 (2009)
28. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering **1**(3), 370–395 (2007)
29. Jensen, A.S., Dignum, V.: AORTA: adding organizational reasoning to agents. In: AAMAS '14, pp. 1493–1494 (2014)
30. Jensen, A.S., Dignum, V., Villadsen, J.: The AORTA architecture: Integrating organizational reasoning in *Jason*. In: EMAS@AAMAS '14, pp. 112–128 (2014)
31. Jones, A.J.I., Sergot, M.: On the characterisation of law and computer systems: The normative systems perspective. In: Deontic Logic in Computer Science: Normative System Specification, pp. 275–307. John Wiley & Sons (1993)
32. Kumar, S., Huber, M.J., Cohen, P.R., Mcgee, D.R.: Toward a formalism for conversation protocols using joint intention theory. Comp. Intelligence **18** (2002)
33. Meneguzzi, F., Luck, M.: Norm-based behaviour modification in BDI agents. In: AAMAS '09, pp. 177–184 (2009)
34. Padgham, L., Lambrix, P.: Formalisations of Capabilities for BDI-Agents. Autonomous Agents and Multi-Agent Systems **10**(3), 249–271 (2005)
35. Plotkin, G.D.: A structural approach to operational semantics. J. Log. Algebr. Program. **60-61**, 17–139 (2004)
36. Ranathunga, S., Cranefield, S., Purvis, M.: Integrating expectation monitoring into bdi agents. In: L. Dennis, O. Boissier, R.H. Bordini (eds.) Programming Multi-Agent Systems, *Lecture Notes in Computer Science*, vol. 7217, pp. 74–91. Springer Berlin Heidelberg (2012)
37. Rao, A.S.: AgentSpeak (L): BDI agents speak out in a logical computable language. Agents Breaking Away (L) (1996)
38. Rao, A.S., Georgeff, M.P.: BDI Agents: From Theory to Practice. In: ICMAS '95 (1995)
39. van Riemsdijk, M.B., Dignum, V., Jonker, C.M., Aldewereld, H.: Programming Role Enactment through Reflection. In: 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, vol. 2, pp. 133–140. IEEE Computer Society (2011)

40. Riemsdijk, M.B., Hindriks, K., Jonker, C.: Programming organization-aware agents. In: ESAW '09. Springer (2009)
41. van der Torre, L.: Contextual deontic logic: Normative agents, violations and independence. Annals of Mathematics and Artificial Intelligence pp. 33–63 (2003)
42. van der Torre, L., Tan, Y.H.: Contrary-to-duty reasoning with preference-based dyadic obligations. Annals of Mathematics and Artificial Intelligence **27**(1–4), 49–78 (1999)
43. Wallace, I., Rovatsos, M.: A computational framework for practical social reasoning. Computational Intelligence **31**(1), 69–105 (2015). DOI 10.1111/coin.12014. URL http://dx.doi.org/10.1111/coin.12014